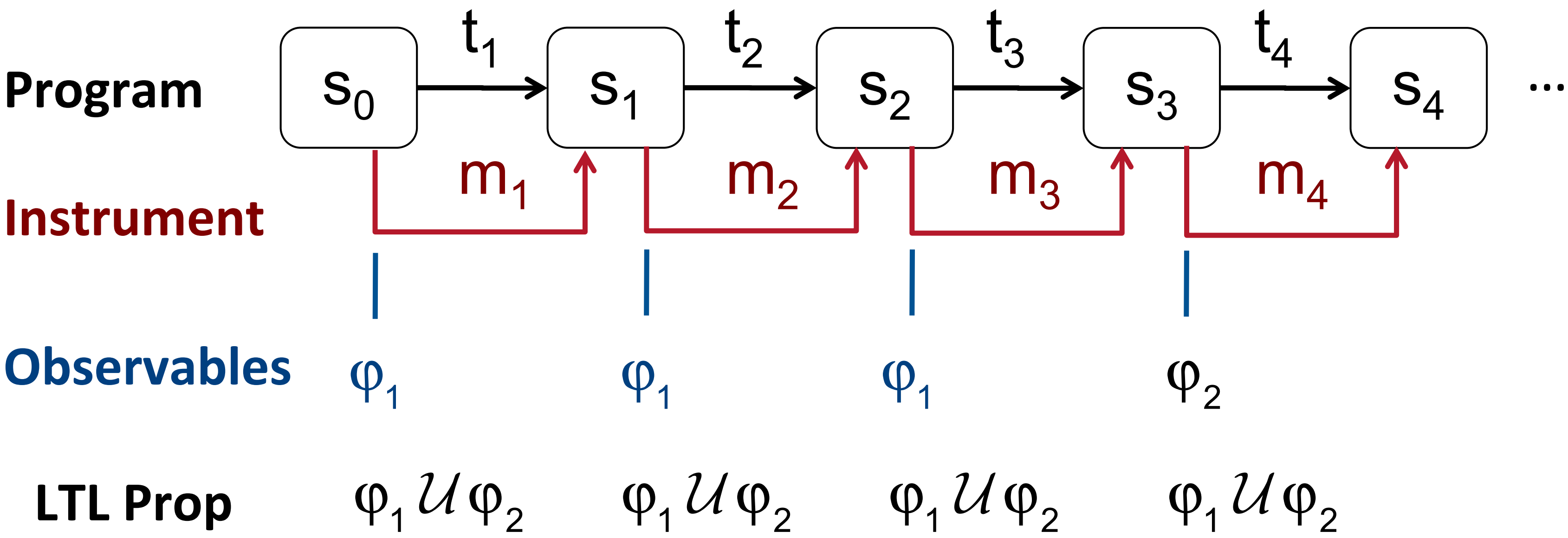
Adaptive Runtime Verification

Radu Grosu
Vienna University of Technology

Joint work with:

E. Bartocci, S. Callanan, K. Havelund, K. Kalajdzik
S.A. Smolka, S.D. Stoller, J. Seyster, E. Zadok

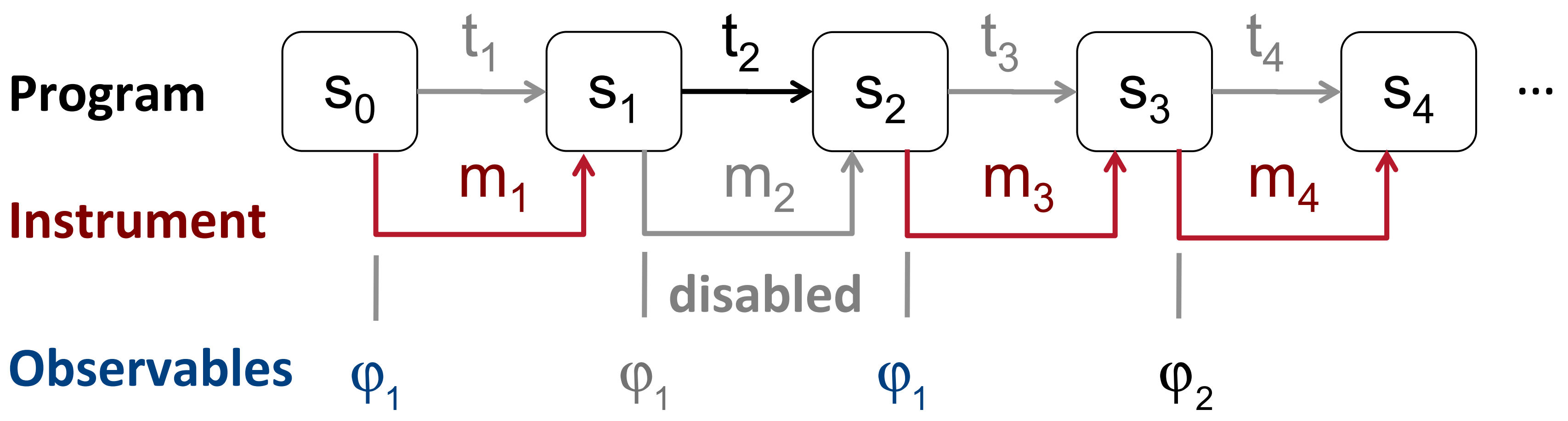
Runtime Verification Problem



Why RV? Most properties cannot be checked statically

Problem: It introduces overhead!

Overhead Control Problem



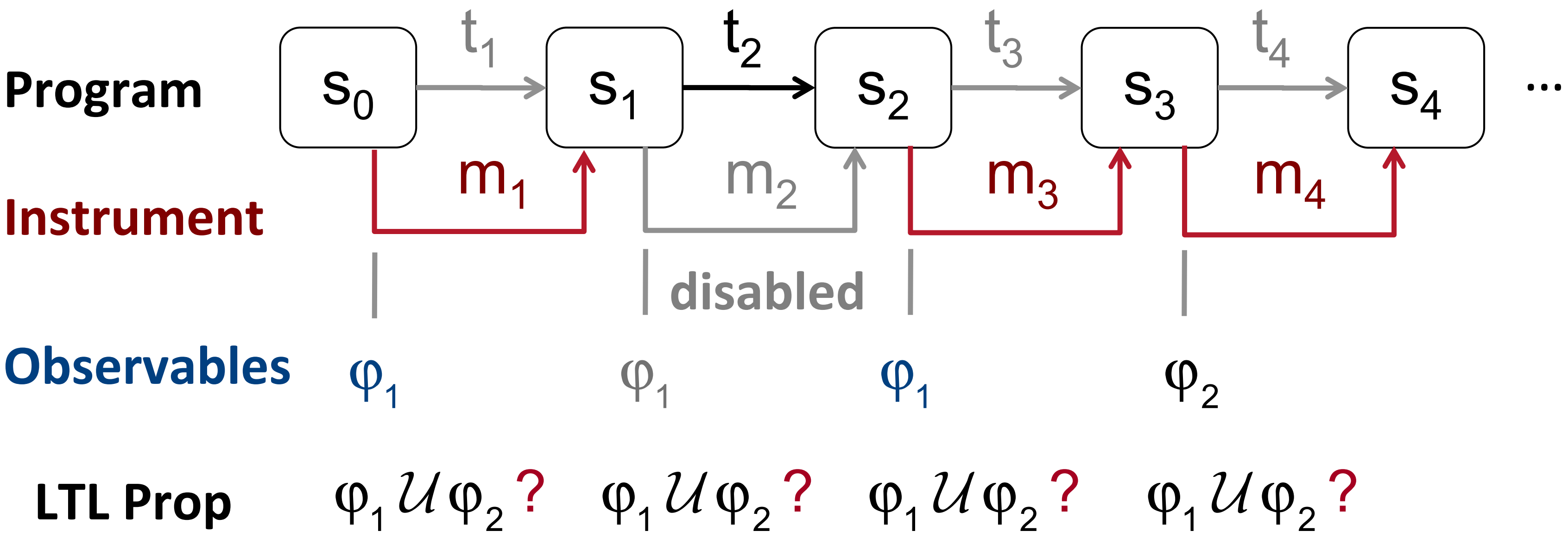
Solution: Use a PID controller

LTL Prop $\varphi_1 \mathcal{U} \varphi_2$? $\varphi_1 \mathcal{U} \varphi_2$? $\varphi_1 \mathcal{U} \varphi_2$? $\varphi_1 \mathcal{U} \varphi_2$?

Problem: We do not know anymore if the LTL property holds!

Fix: Learn and Estimate state with a Hidden Markov Model

Stateful Overhead Control



Adaptive RV: Knowing the state one can tune budget towards Instances that are most likely to violate the LTL property next.

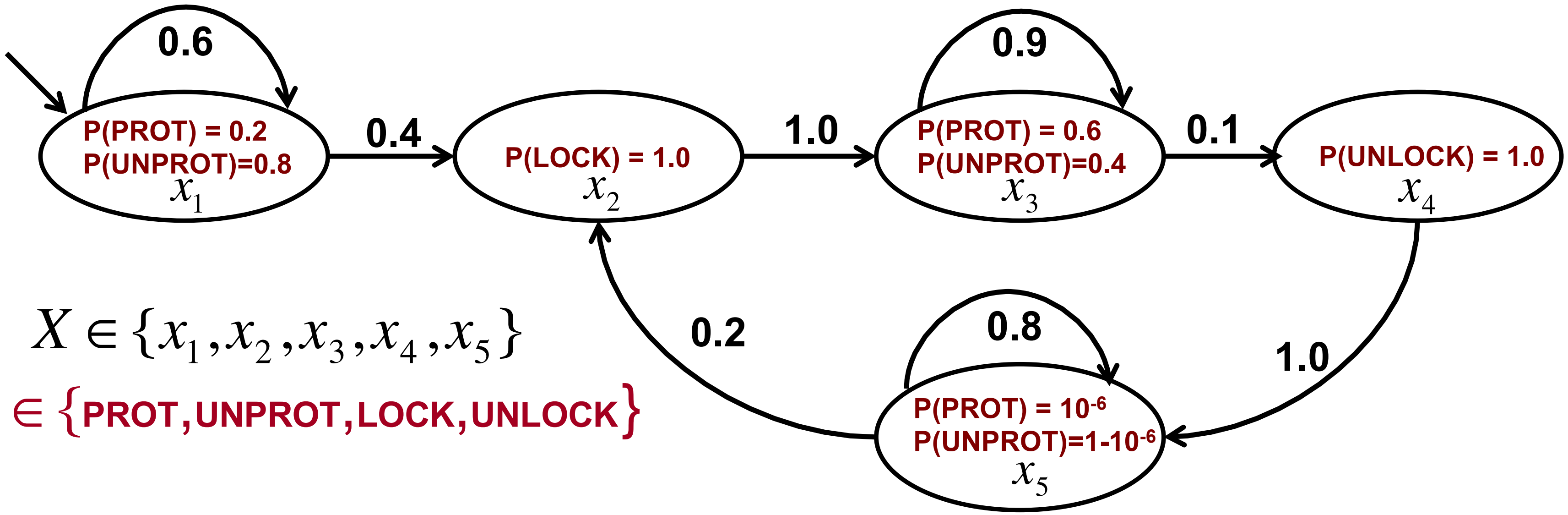
Outline

- Program Model: Hidden Markov Model
- Property: Deterministic Finite Automaton
- Correponding Dynamic Bayesian Network
- State Estimation with Particle Filters
- Comparison of State-Estimation Techniques
- Adaptive Runtime Verification

Program Model: Hidden Markov Model

T_H	x_1	x_2	x_3	x_4	x_5
x_1	0.6	0.4			
x_2			1		
x_3			0.9	0.1	$P(x_5 \mid x_3)$
x_4					1
x_5		0.2			0.8

O_H	PROT	UNPROT	LOCK	UNLOCK
x_1	0.2	0.8		
x_2			1	
x_3	0.6	0.4		$P(\text{UNLOCK} \mid x_3)$
x_4				1
x_5	10^{-6}	$1-10^{-6}$		



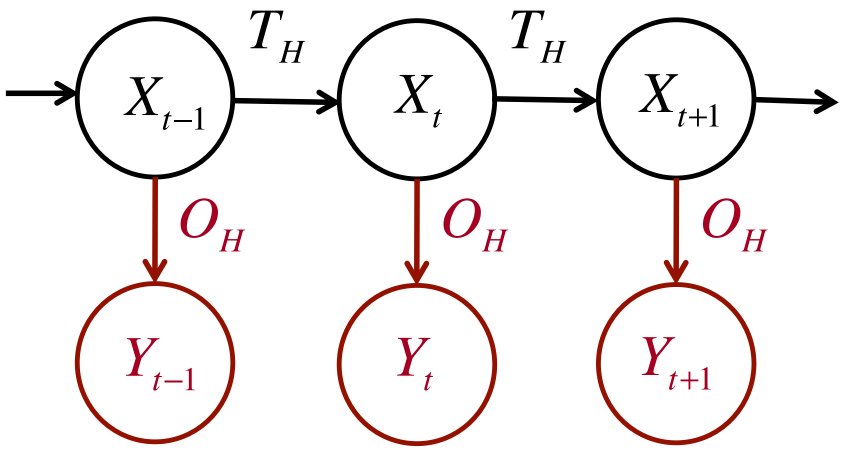
$$X \in \{x_1, x_2, x_3, x_4, x_5\}$$

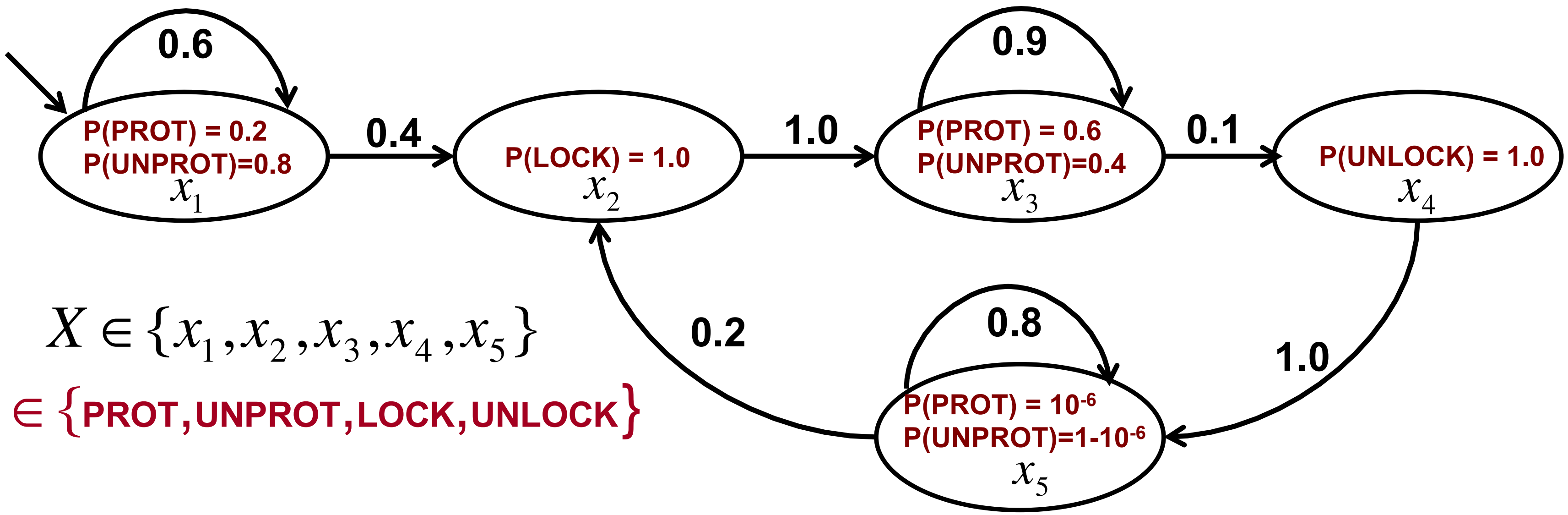
$$Y \in \{\text{PROT}, \text{UNPROT}, \text{LOCK}, \text{UNLOCK}\}$$

Program Model: Hidden Markov Model

$$P(X_{t+1} \mid X_t) = T_H$$

$$P(Y_t \mid X_t) = O_H$$

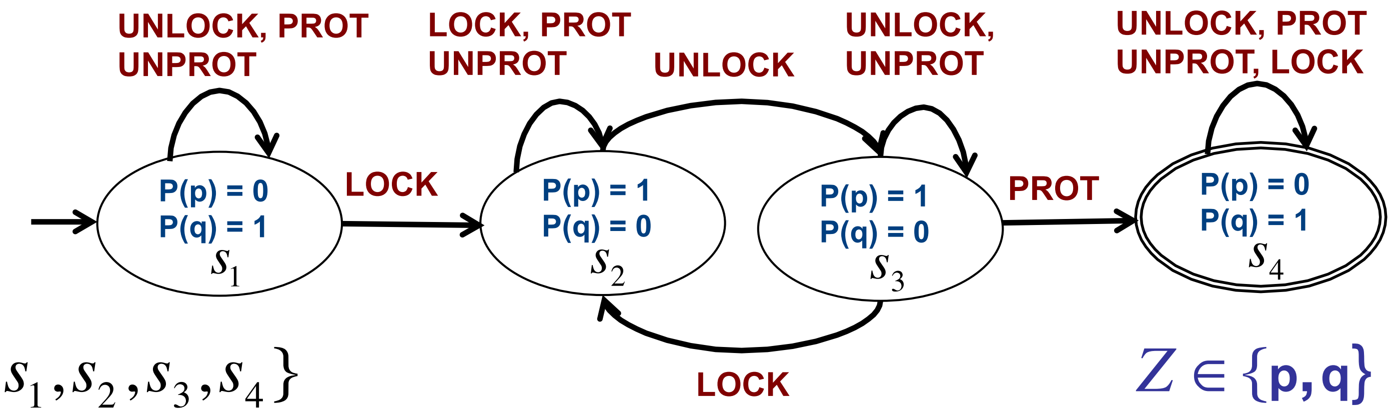




$$X \in \{x_1, x_2, x_3, x_4, x_5\}$$

$$Y \in \{\text{PROT}, \text{UNPROT}, \text{LOCK}, \text{UNLOCK}\}$$

Property: DFA



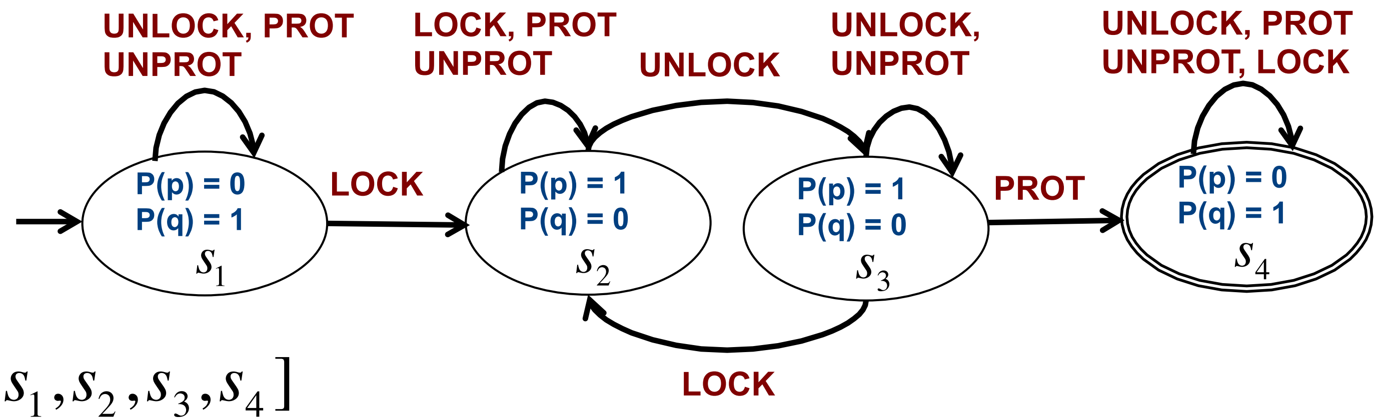
$S \in \{s_1, s_2, s_3, s_4\}$

$Z \in \{\mathbf{p}, \mathbf{q}\}$

T_D	s_1	s_2	s_3	s_4
s_1	PROT UNPROT UNLOCK	LOCK		
s_2		PROT UNPROT LOCK	UNLOCK	
s_3		LOCK	UNPROT UNLOCK	$P(s_4 \mid s_3, \text{PROT})$
s_4				PROT LOCK UNPROT UNLOCK

O_D	p	q
s_1	0	1
s_2	1	0
s_3	1	$P(\mathbf{q} \mid s_3)$
s_4	0	1

Property: DFA

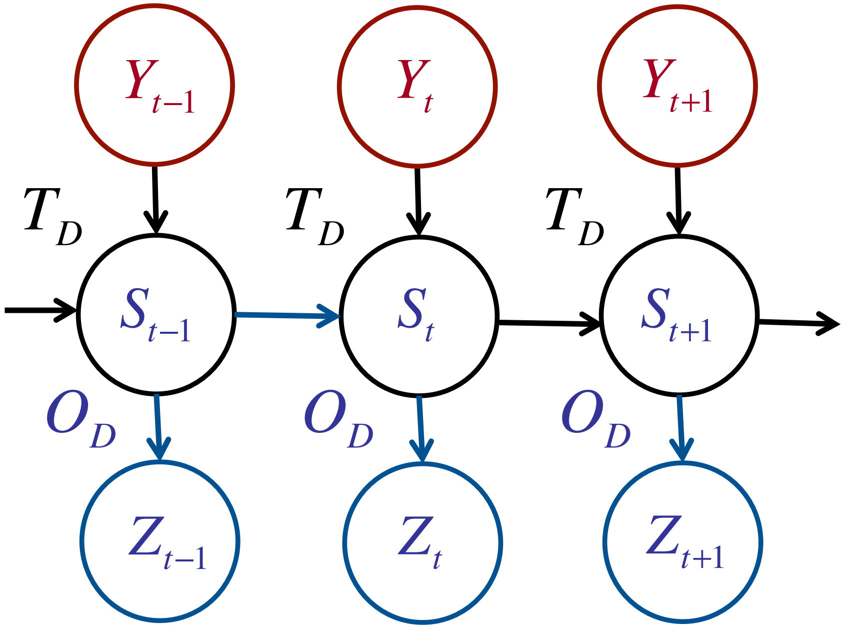


$$S = [s_1, s_2, s_3, s_4]$$

$$Z \in \{\mathbf{p}, \mathbf{q}\}$$

$$\boldsymbol{P}(S_{t+1} \mid S_t, Y_{t+1}) = T_D$$

$$\boldsymbol{P}(Z_t \mid S_t) = O_D$$



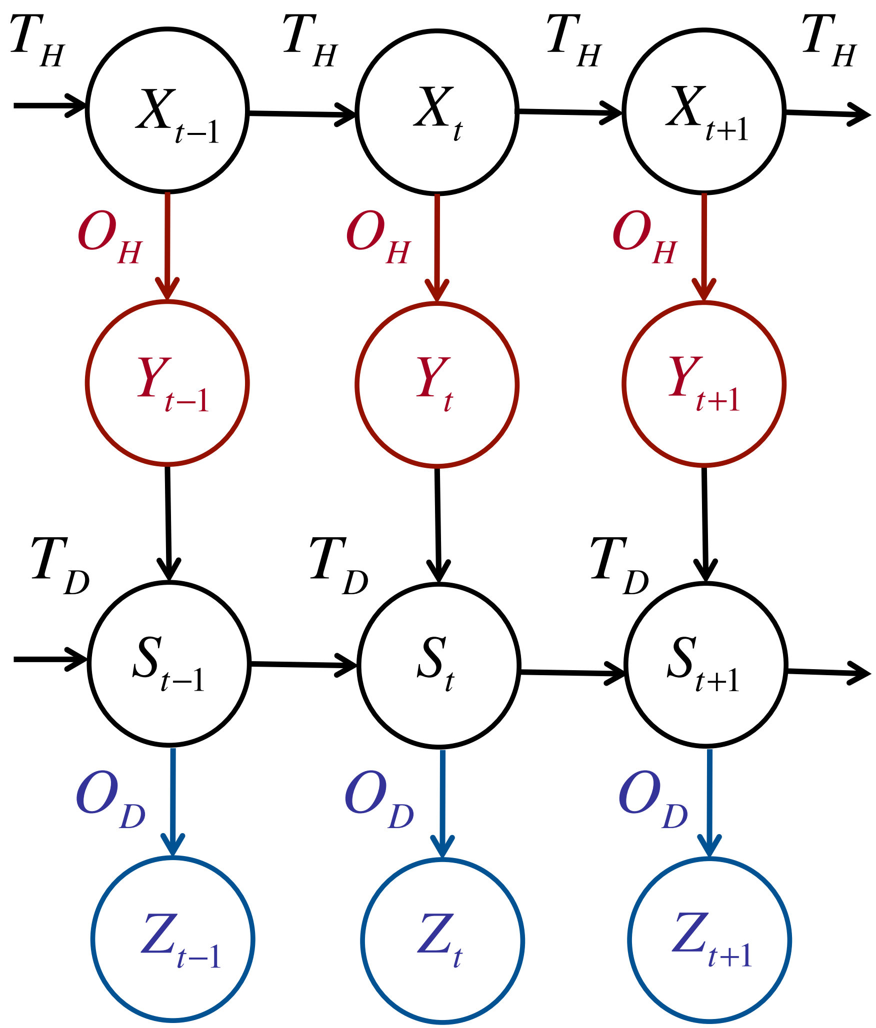
Filtering in DBNs

Given

- $\boldsymbol{P}(X_{t+1} \mid X_t) = T_H$, $P(X_1)$
- $\boldsymbol{P}(Y_t \mid X_t) = O_H$
- $\boldsymbol{P}(S_{t+1} \mid S_t, Y_{t+1}) = T_D$, $P(S_1)$
- $\boldsymbol{P}(Z_t \mid S_t) = O_D$
- $y_1 \ldots y_{t+1}$, $z_1 \ldots z_{t+1}$

Find

- $\boldsymbol{P}(S_{t+1} \mid y_1 \ldots y_{t+1}, z_1 \ldots z_{t+1})$



Computing $P(X_{t+1} \mid y_{1:t+1})$

Exact computation (RVSE) :

- $P(X_{t+1} \mid y_{1:t+1})^t = \alpha \, O^d_{y_{t+1}} \, T^t \, P(X_t \mid y_{1:t})^t$
- **Expensive :** Matrix multiplication consumes 80% of time

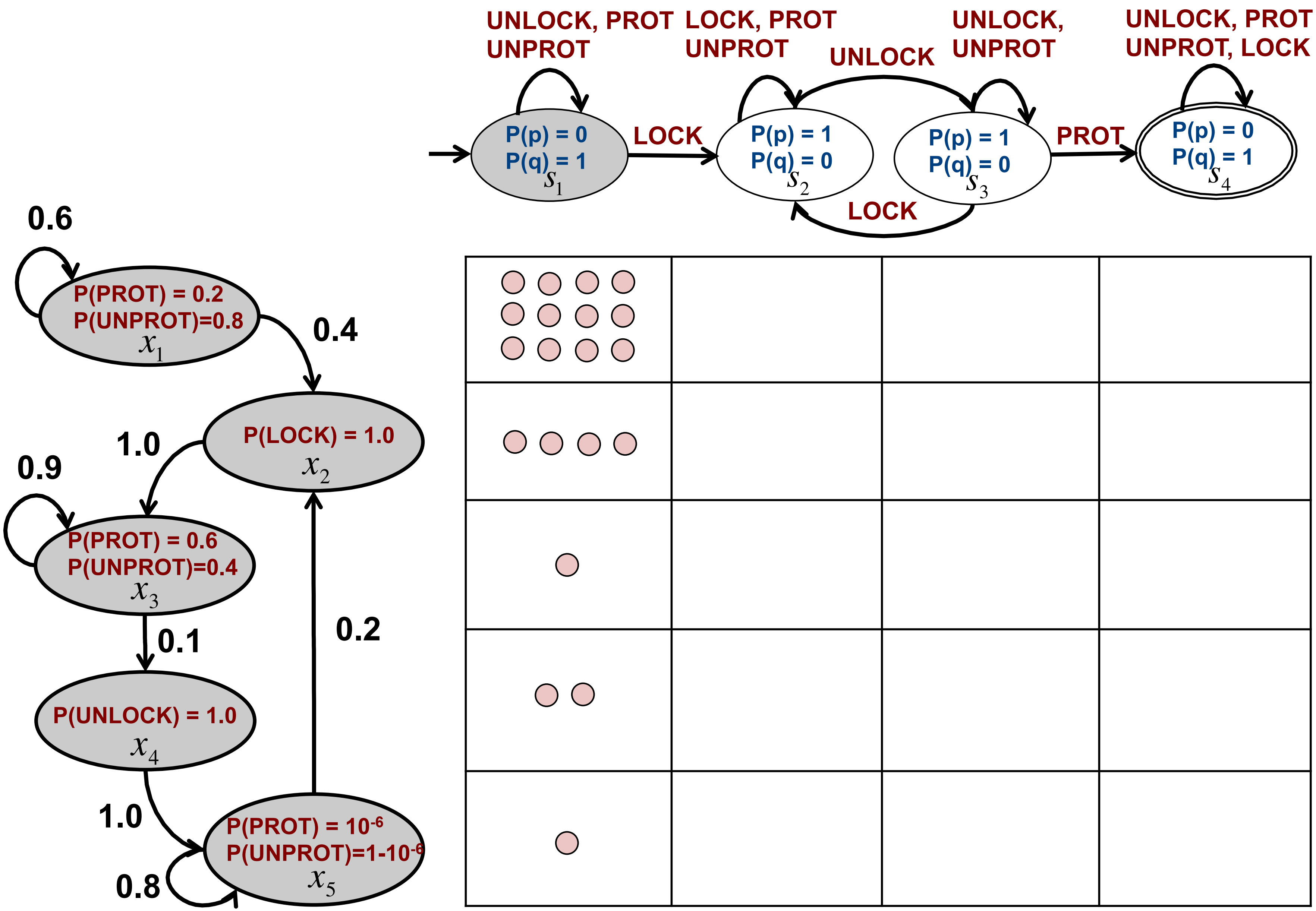
Offline Tabling (AP - RVSE) :

- Approximate computation of $P(X \mid y_{1:*})^t, \forall y_{1:*}$ for error ε
- **Fast :** Memory requirements are very high

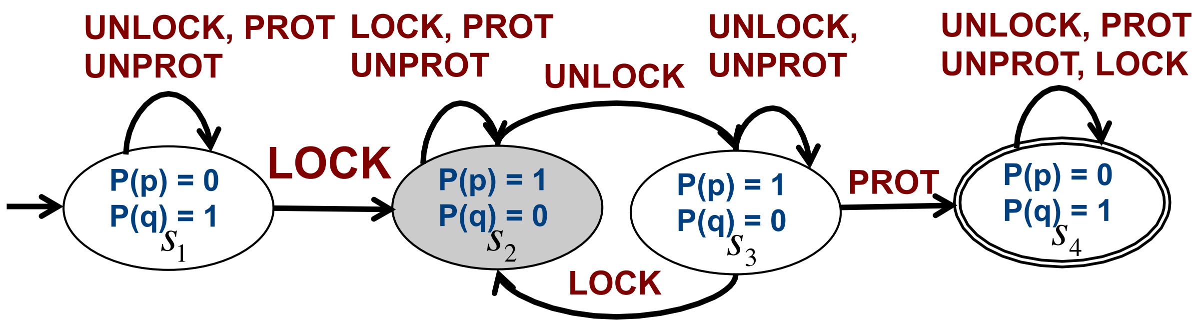
Online Particle Filtering (RVPF) :

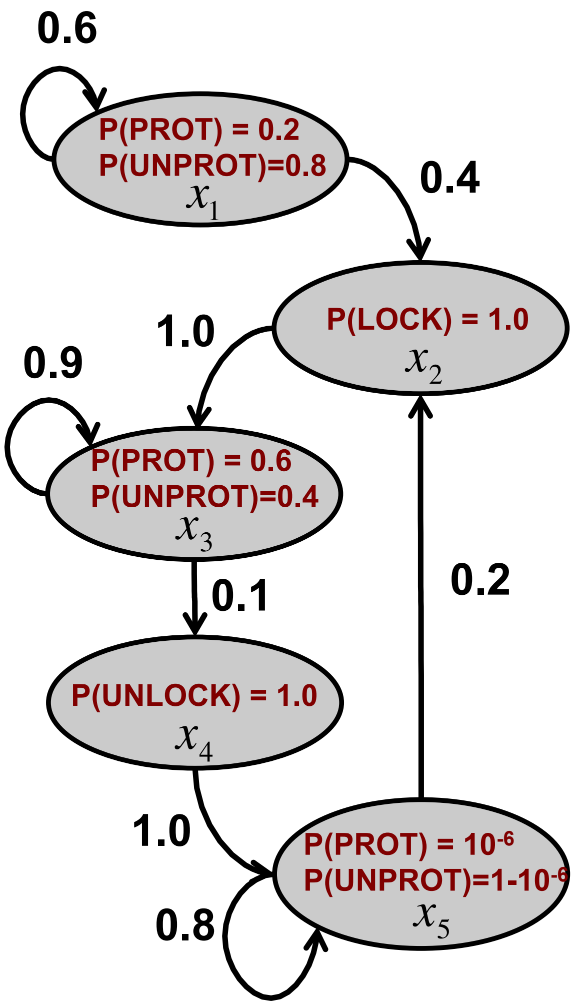
- Approximate computation of $P(X_{t+1} \mid y_{1:t+1})^t$ for n particles
- **Adaptive :** Number n of particles balance overhead / accuracy

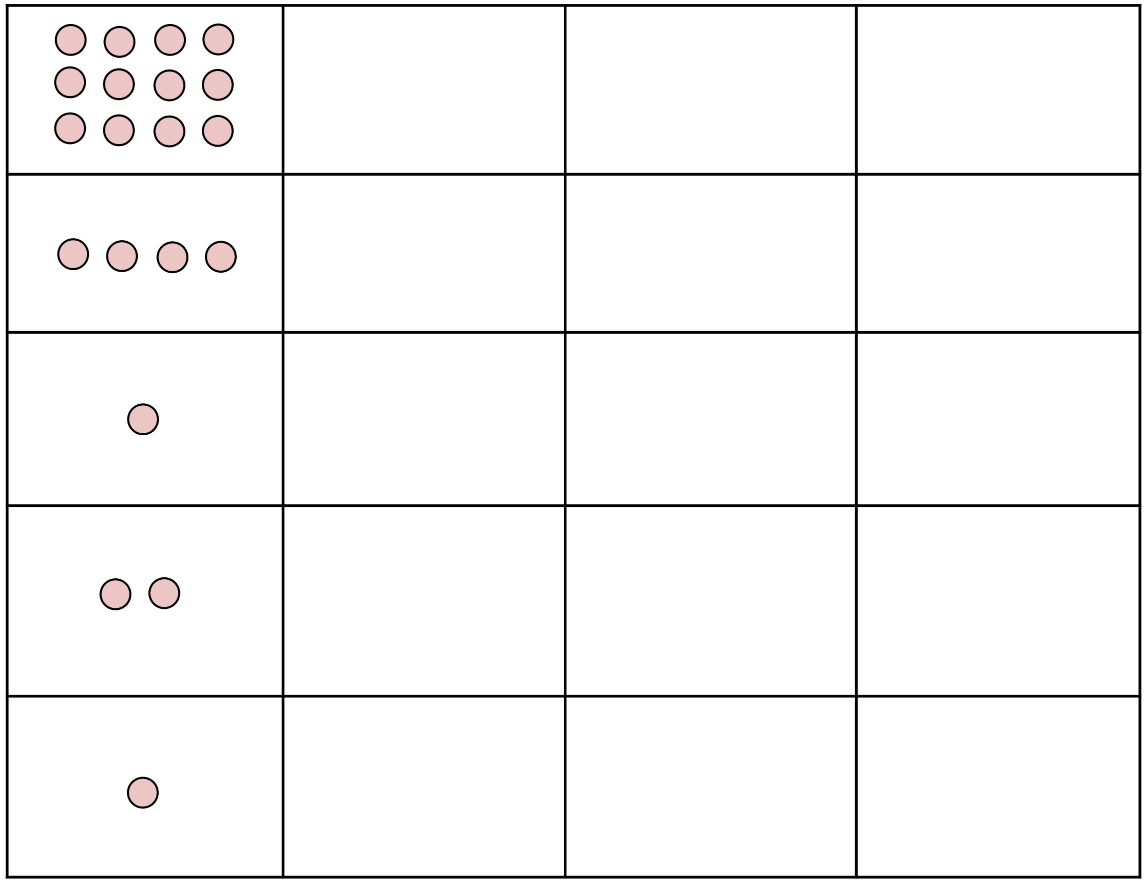
RVPF: Initial Distribution of Particles



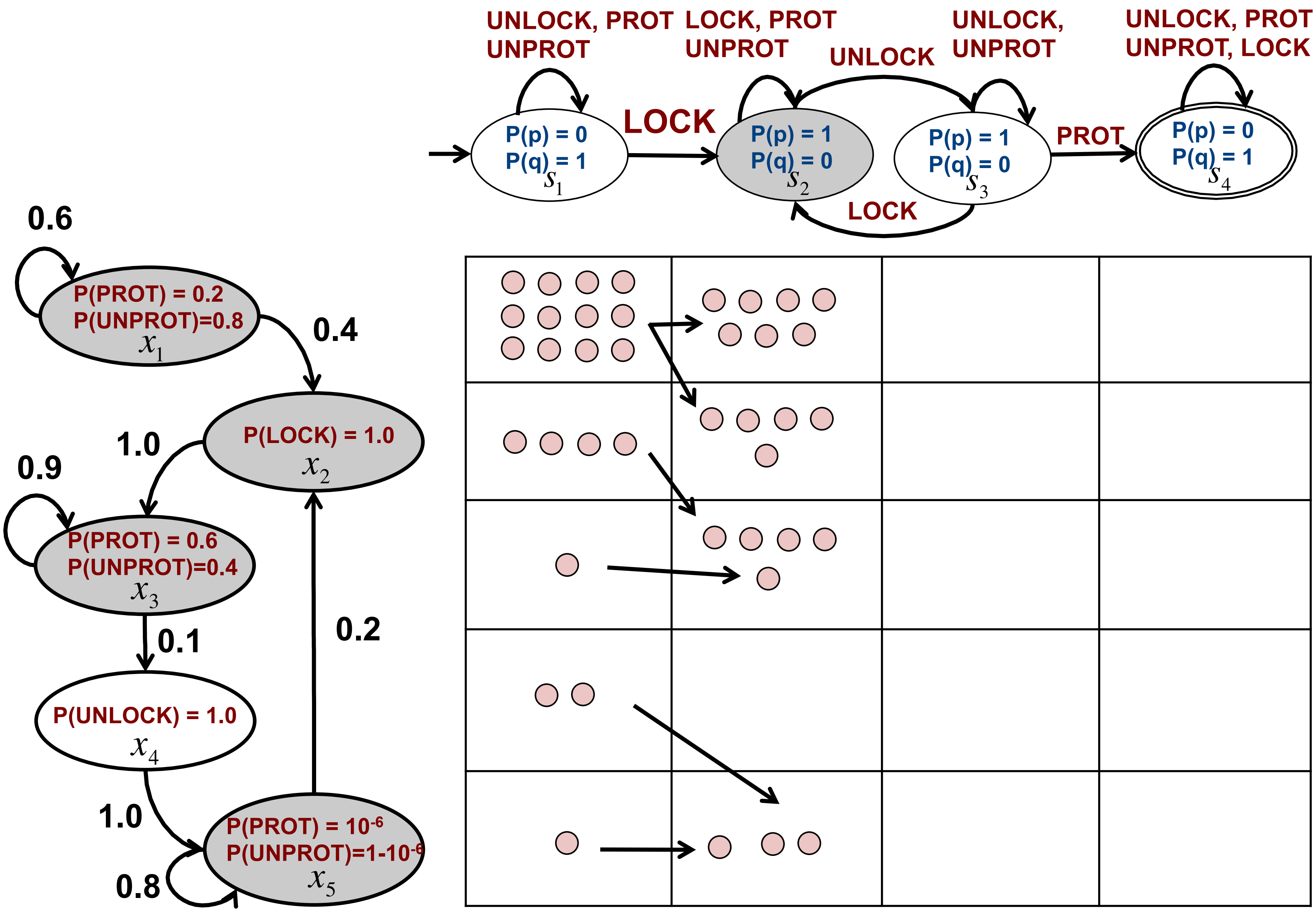
RVPF: Observe LOCK



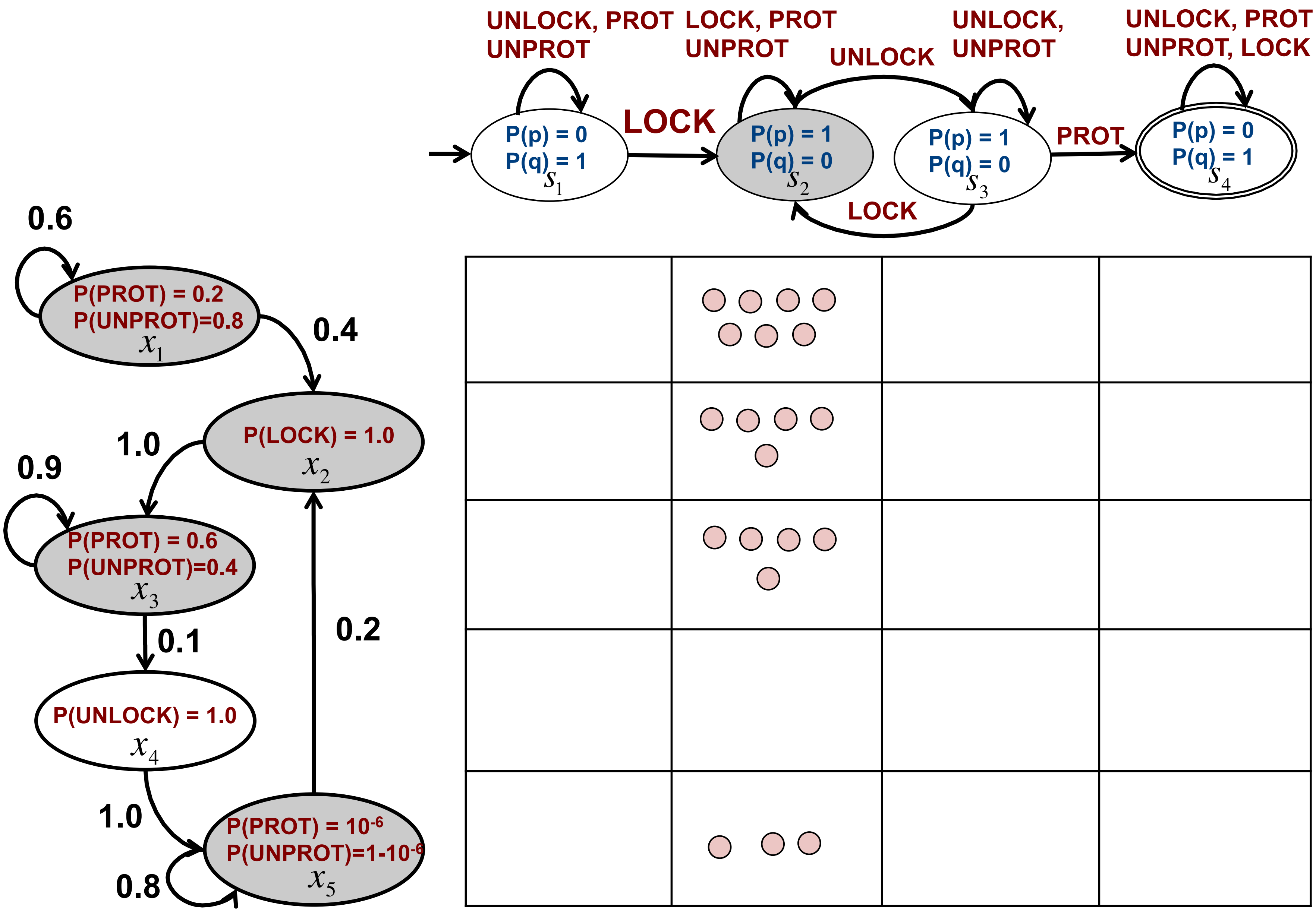




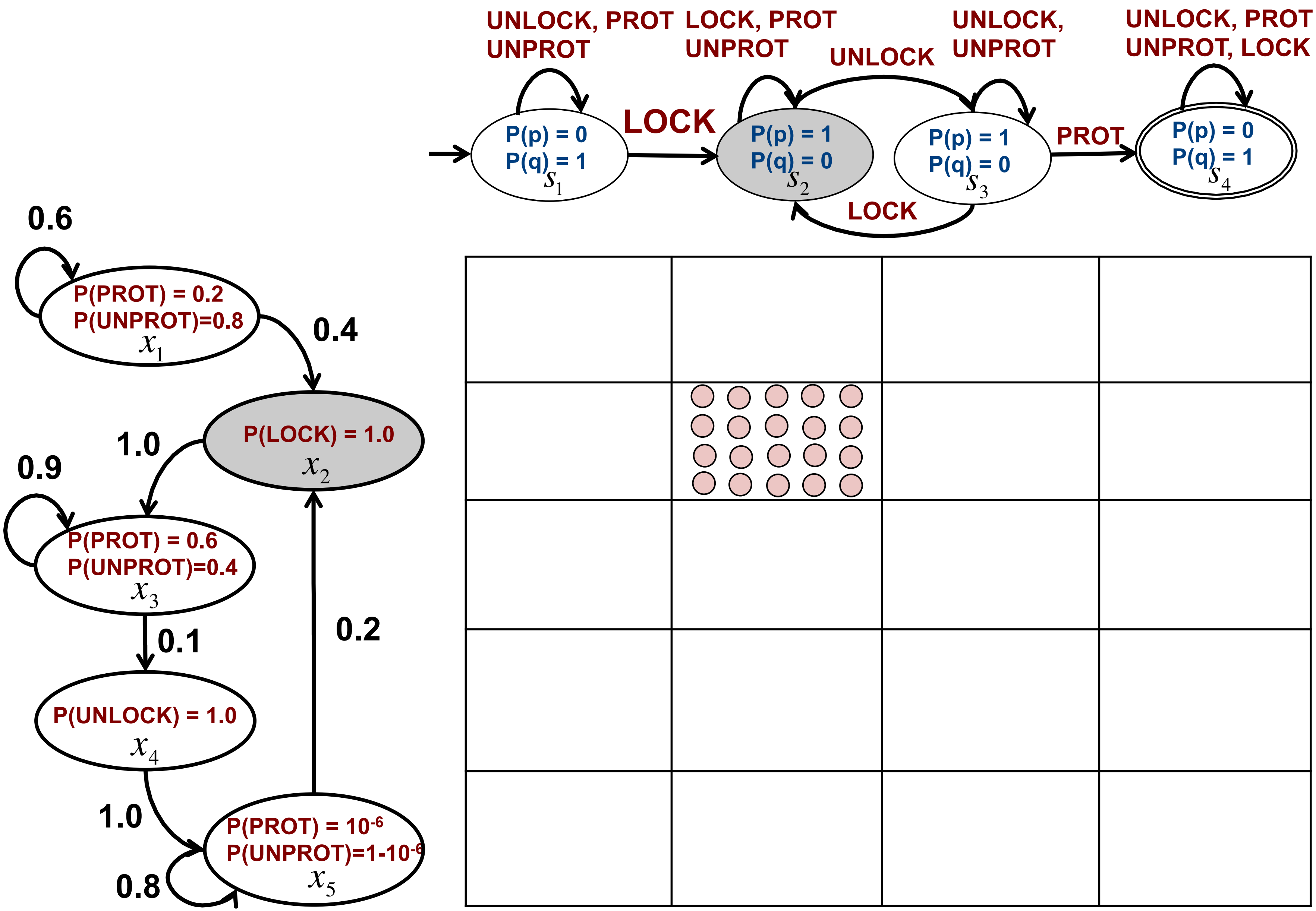
RVPF: Distribute Particles



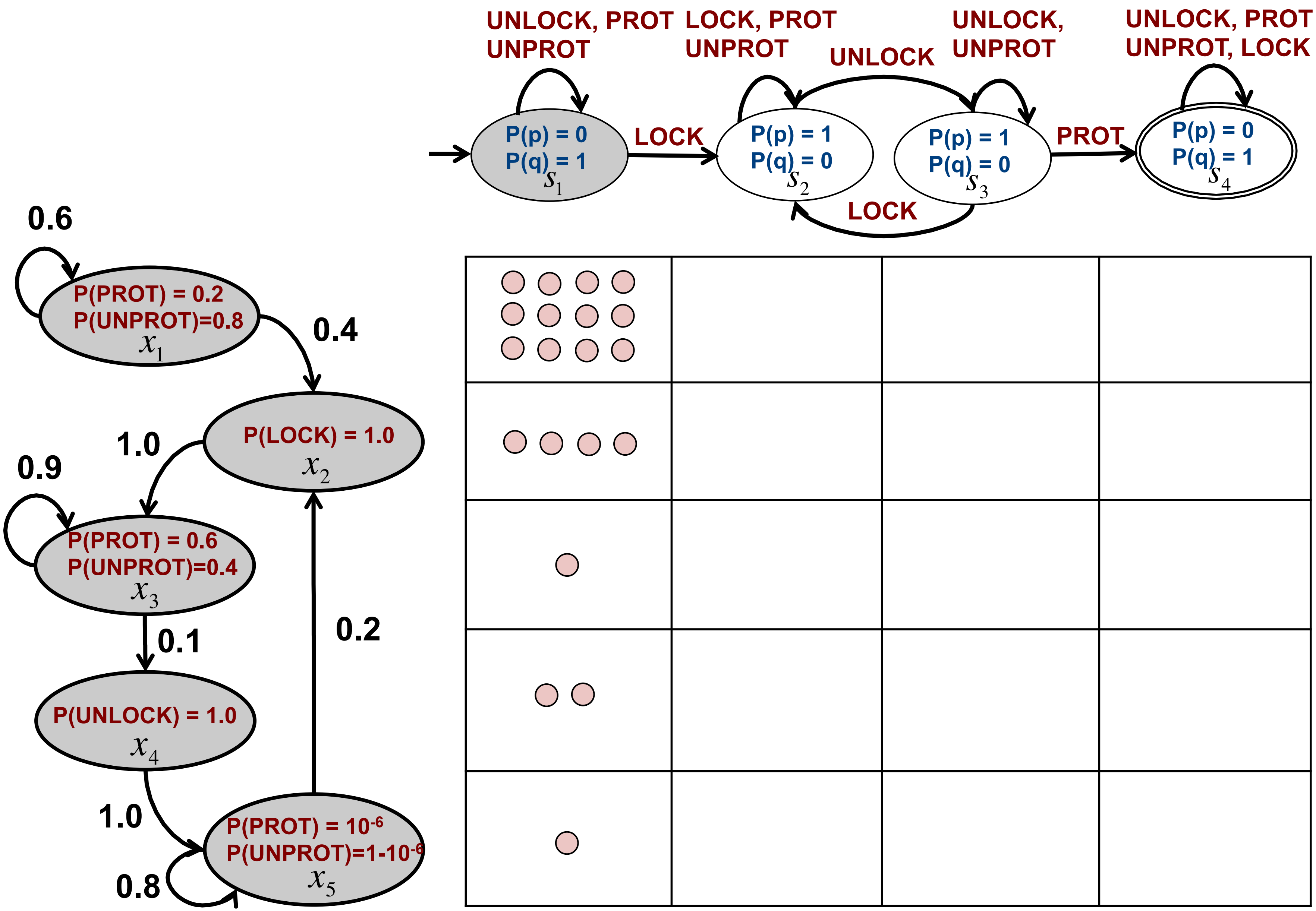
RVPF: New Configuration of Particles



RVPF: Resemple Particles for LOCK

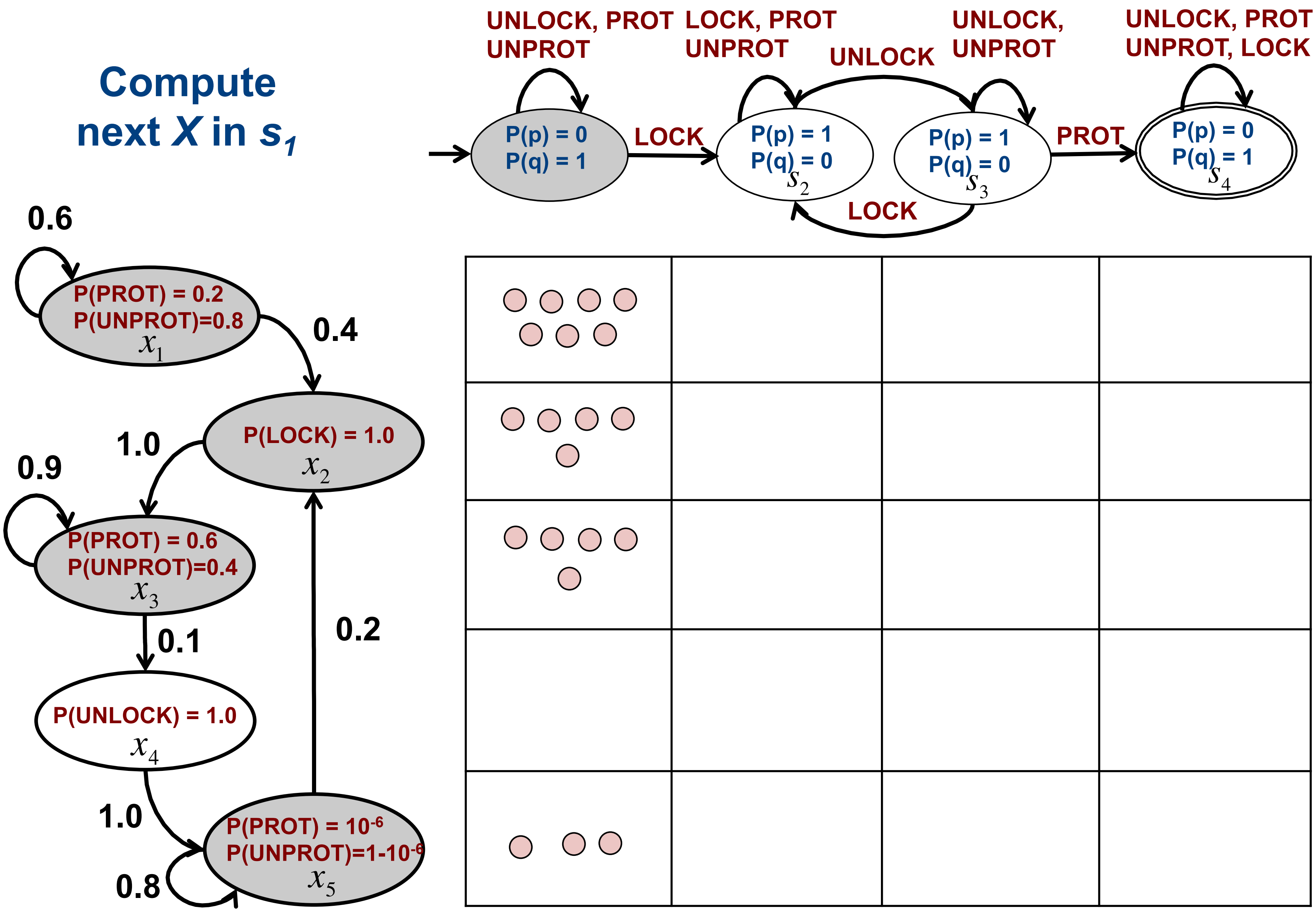


RVPF: Initial distribution of particles

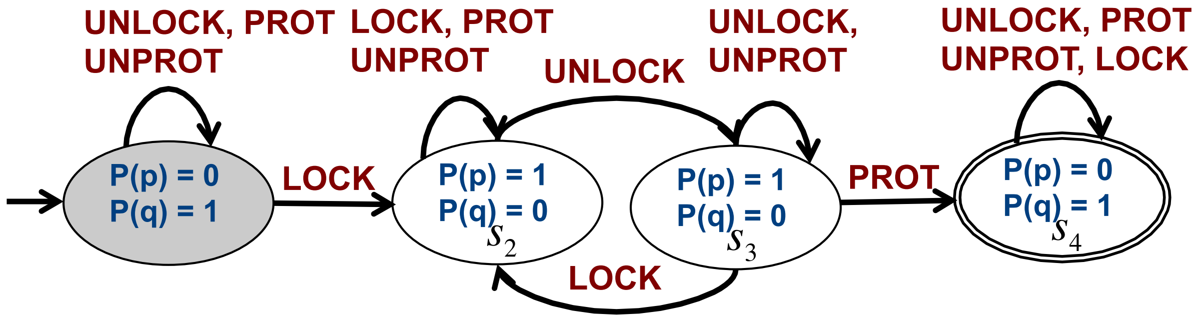


RVPF: Observe GAP

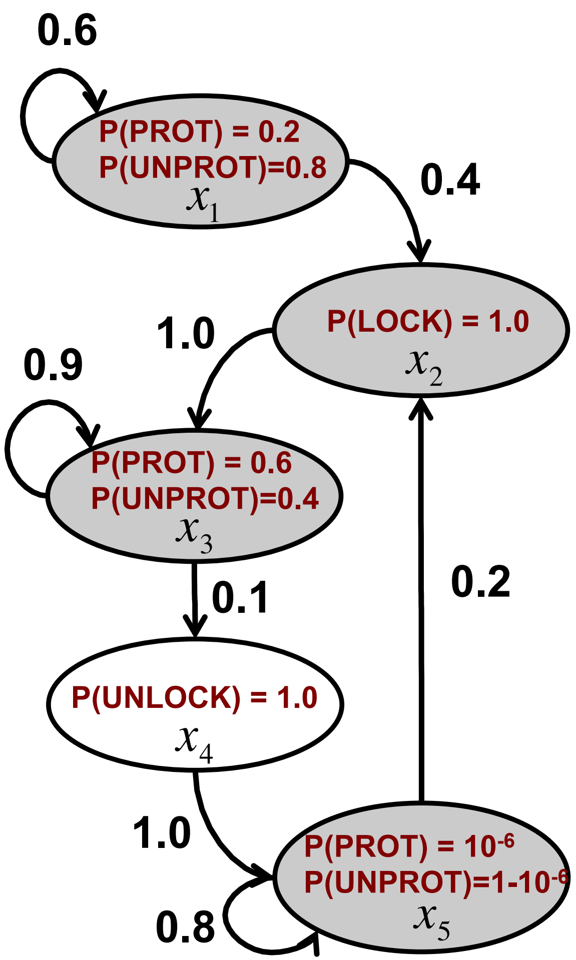
Compute next X in s_1

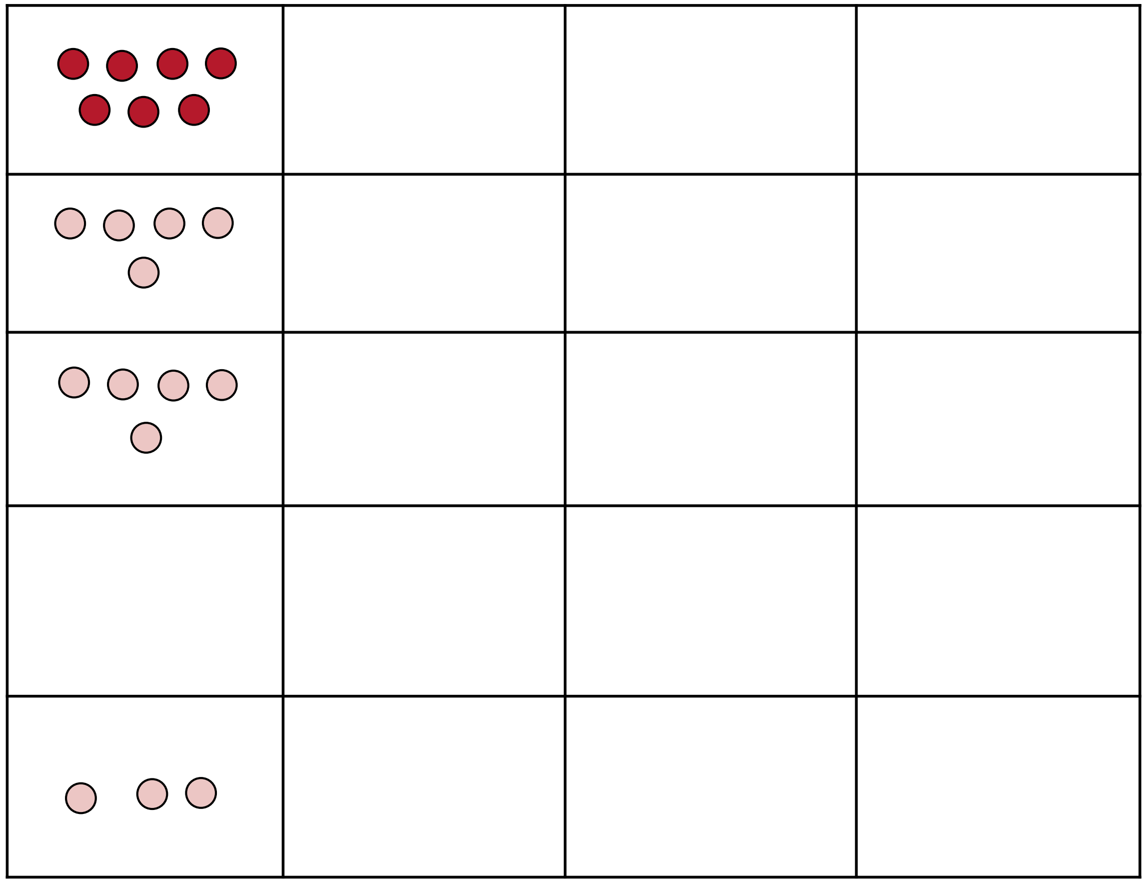


RVPF: Observe GAP



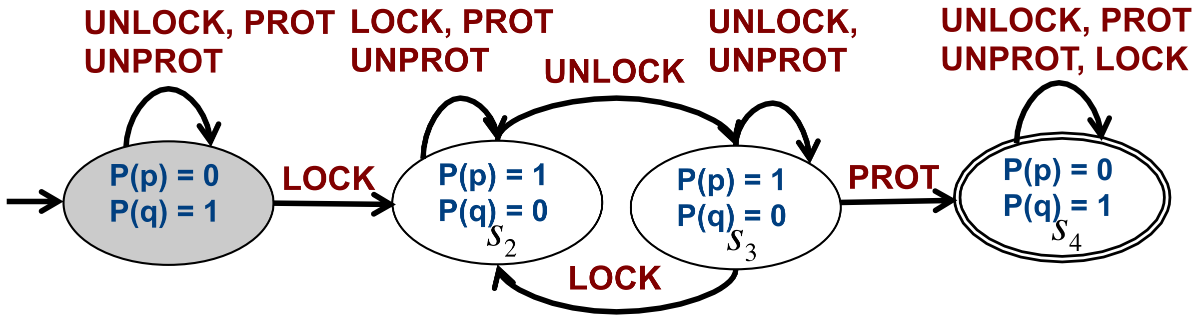
Compute next X in s_1

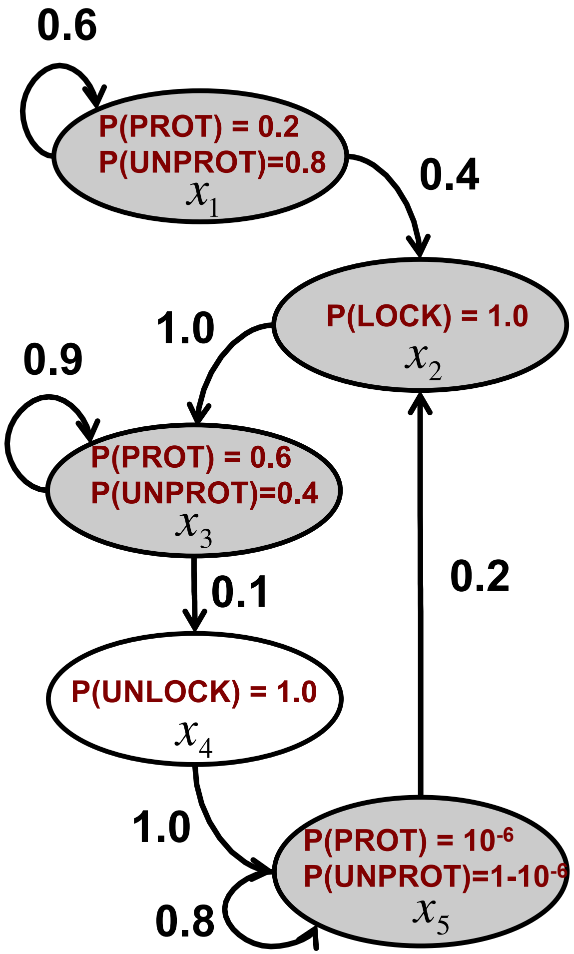


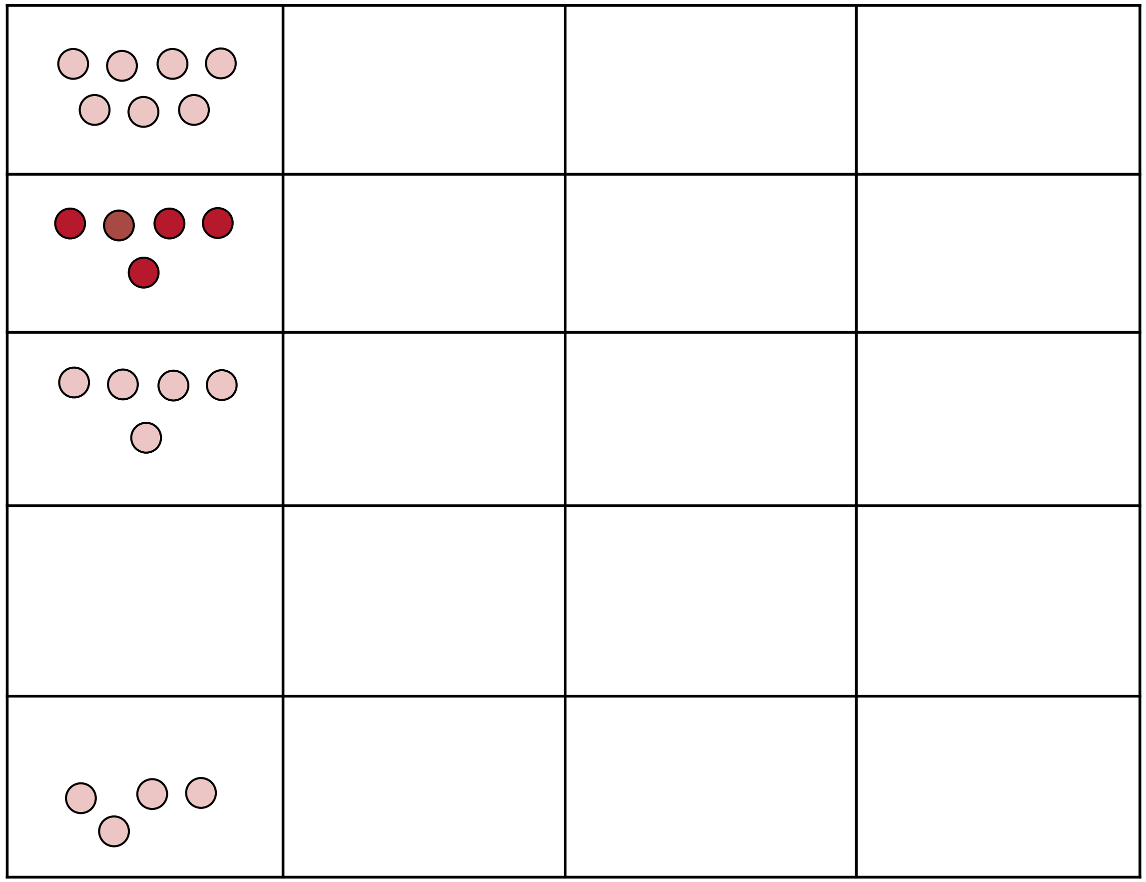


RVPF: Observe GAP

Check O
In every x_i

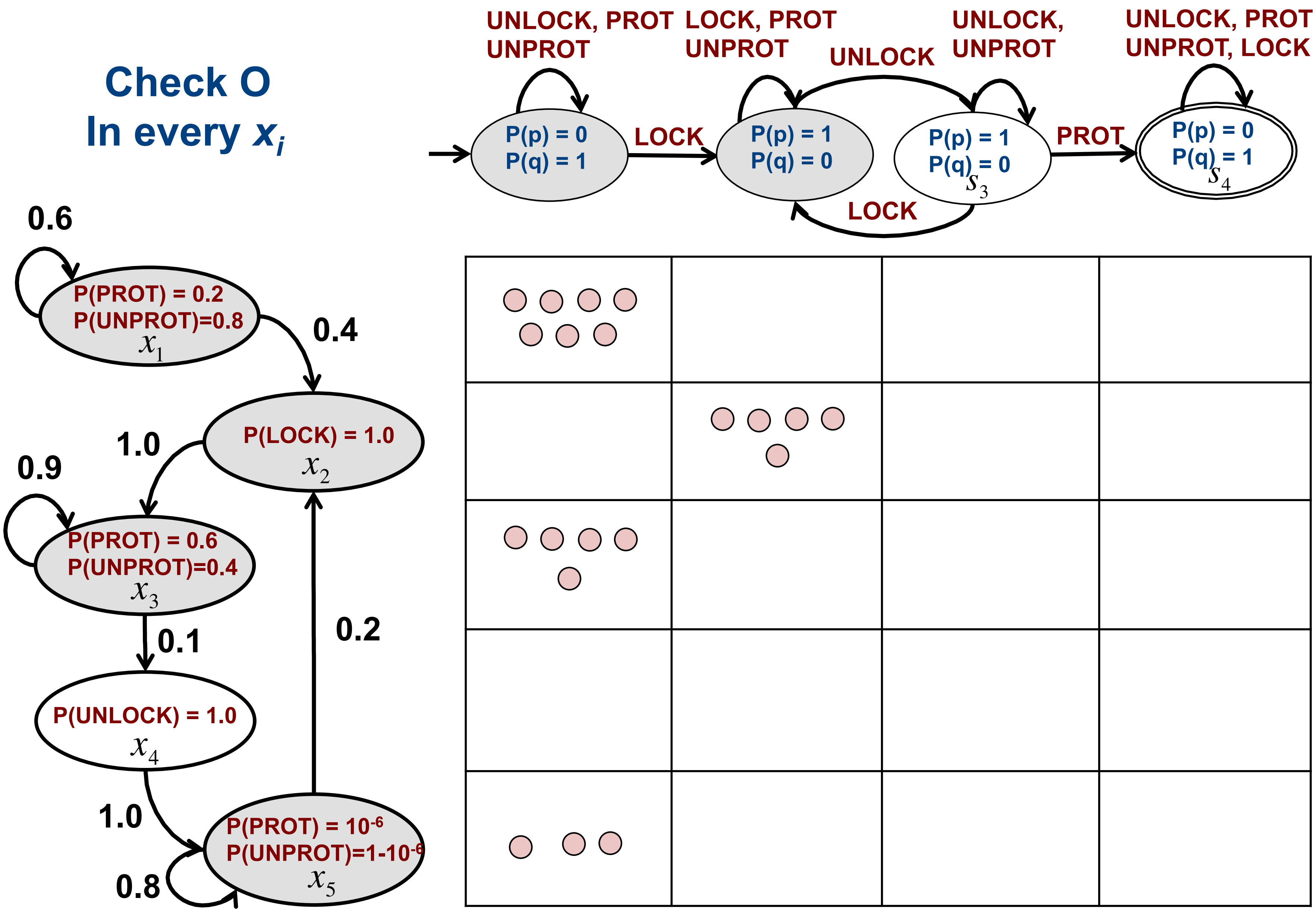




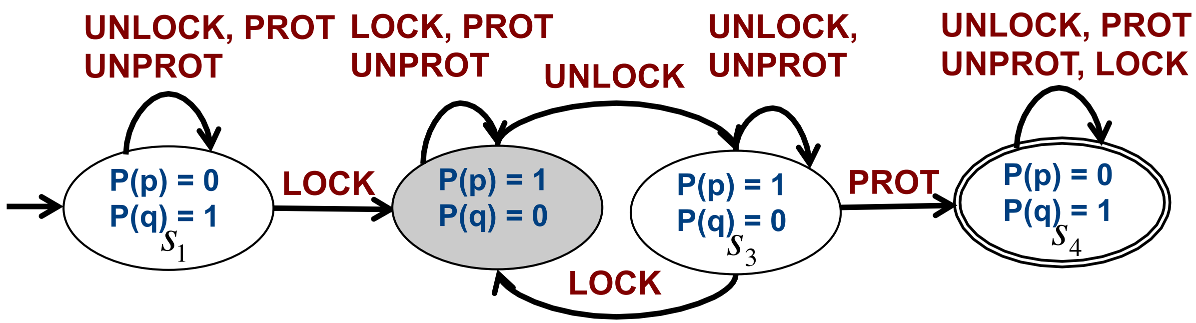


RVPF: Observe GAP

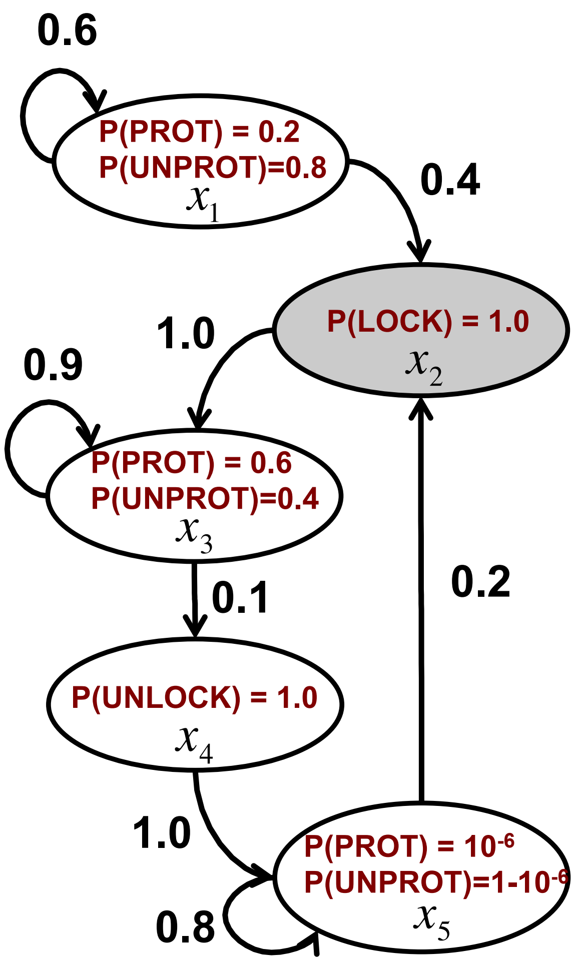
Check O
In every x_i

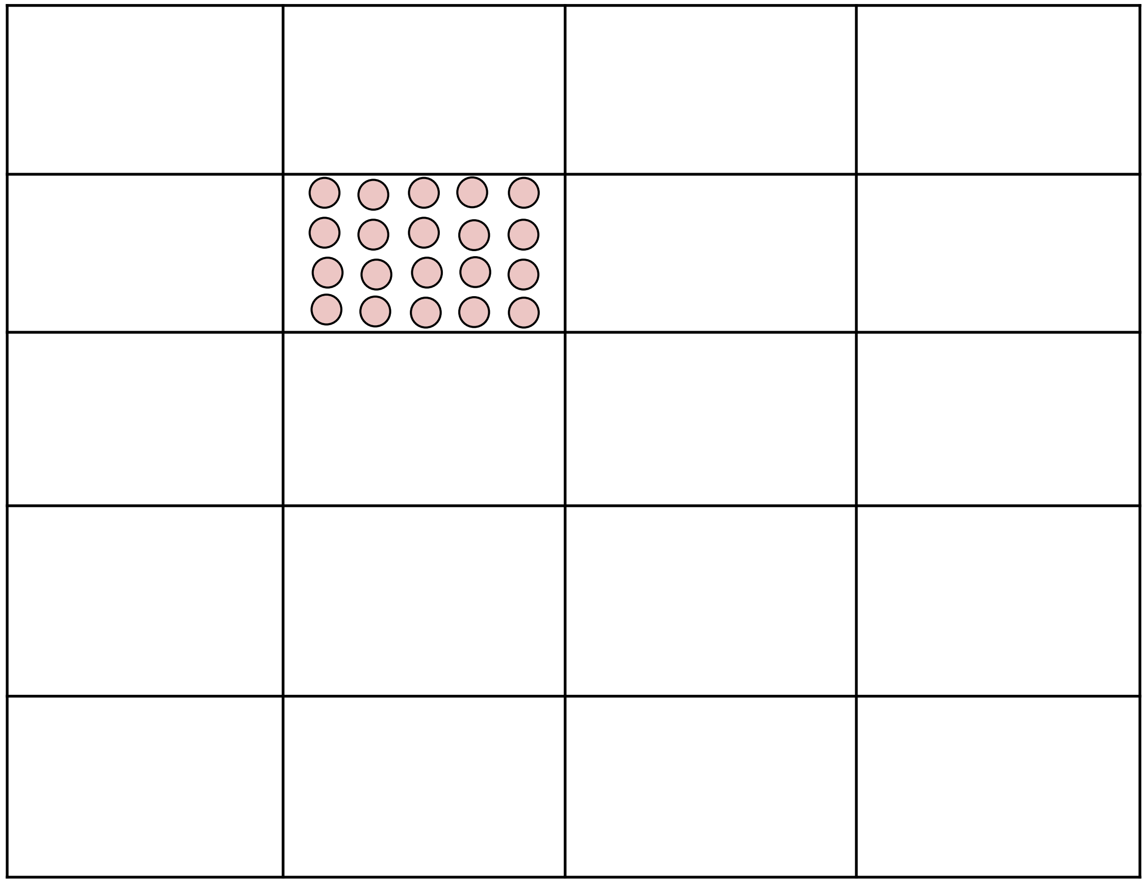


RVPF: Observe a Peek p

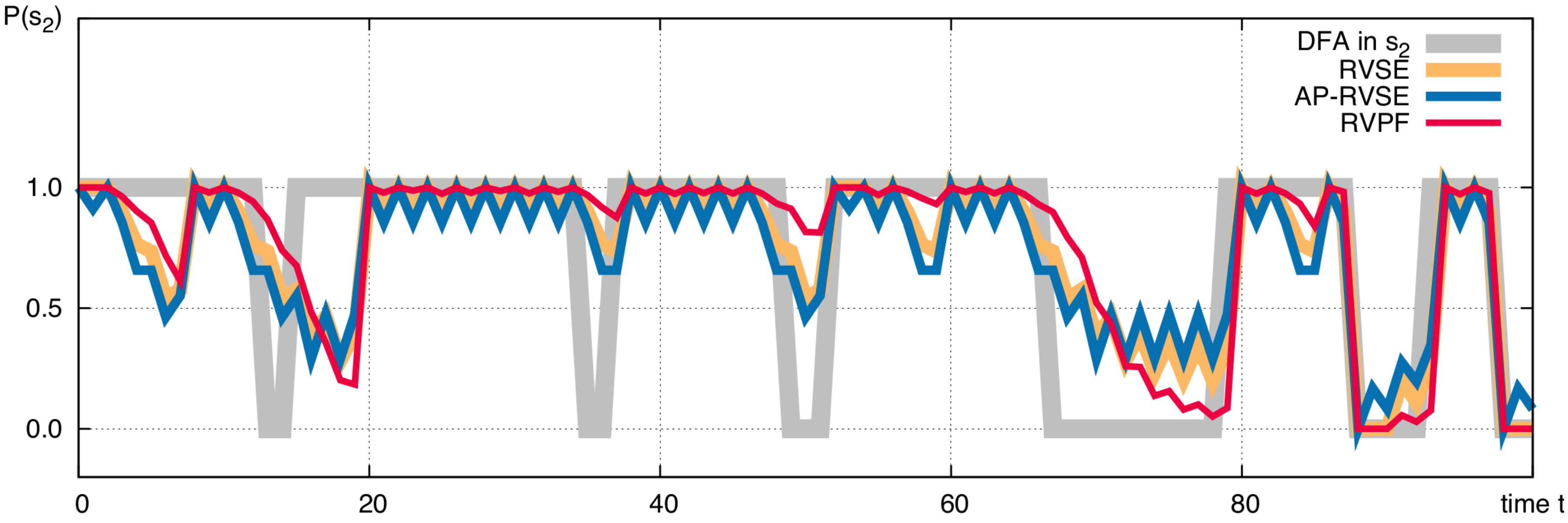


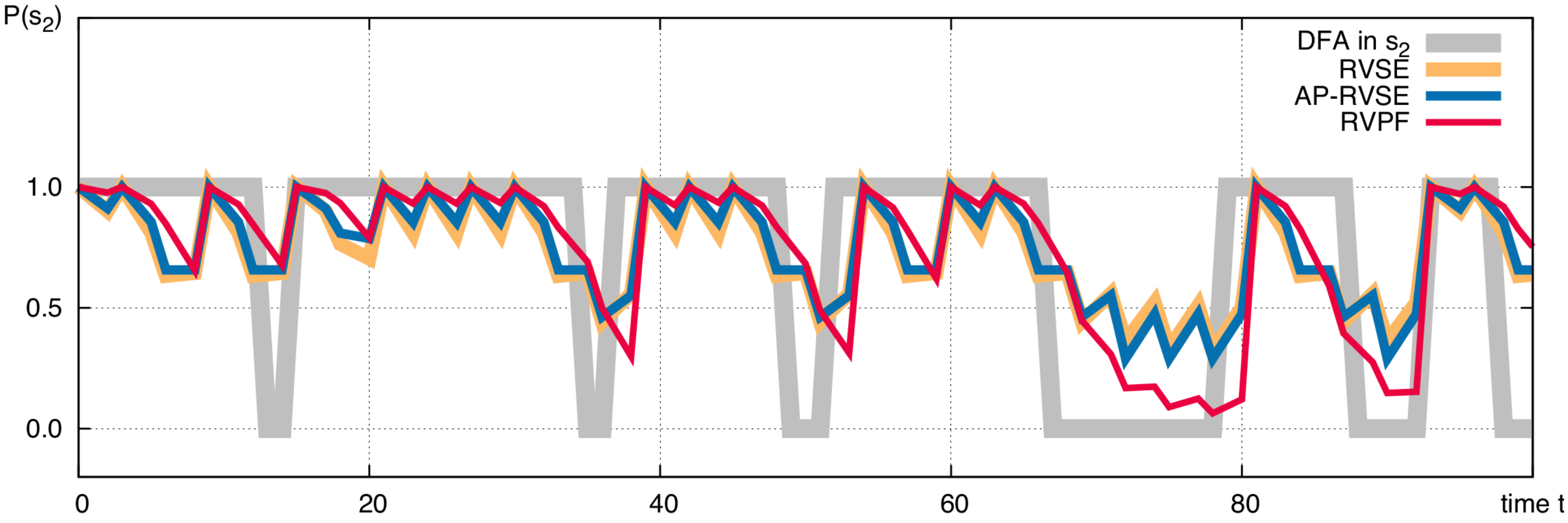
Check O
In every x_i



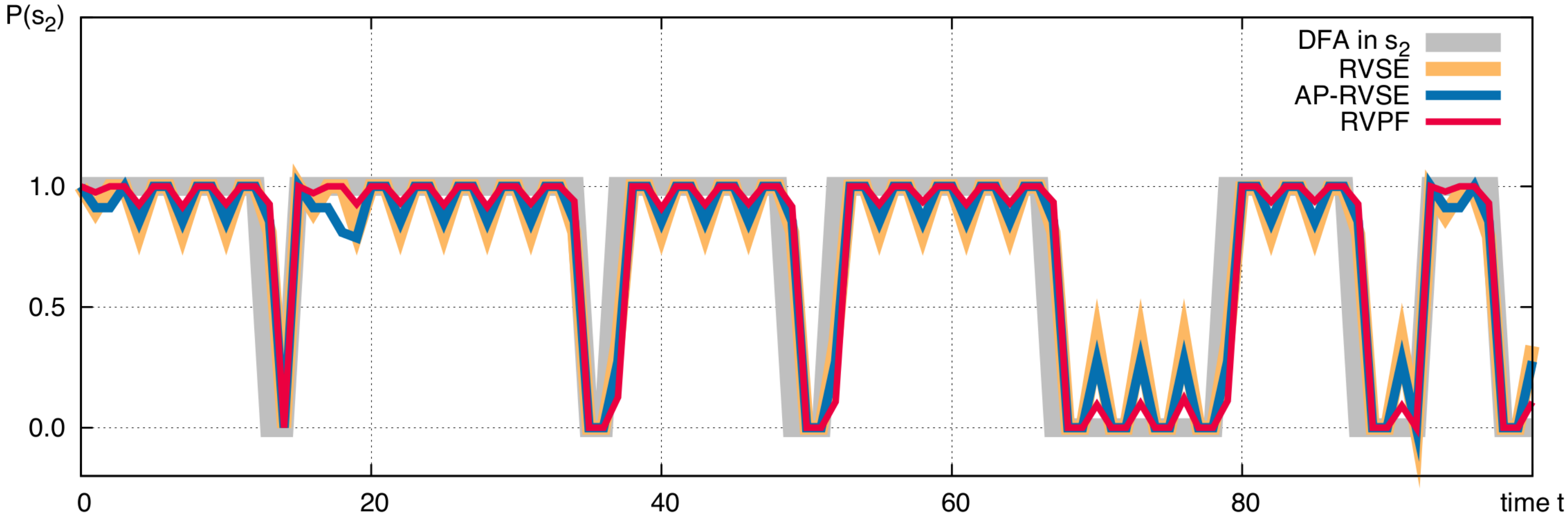


Accuracy Evaluation



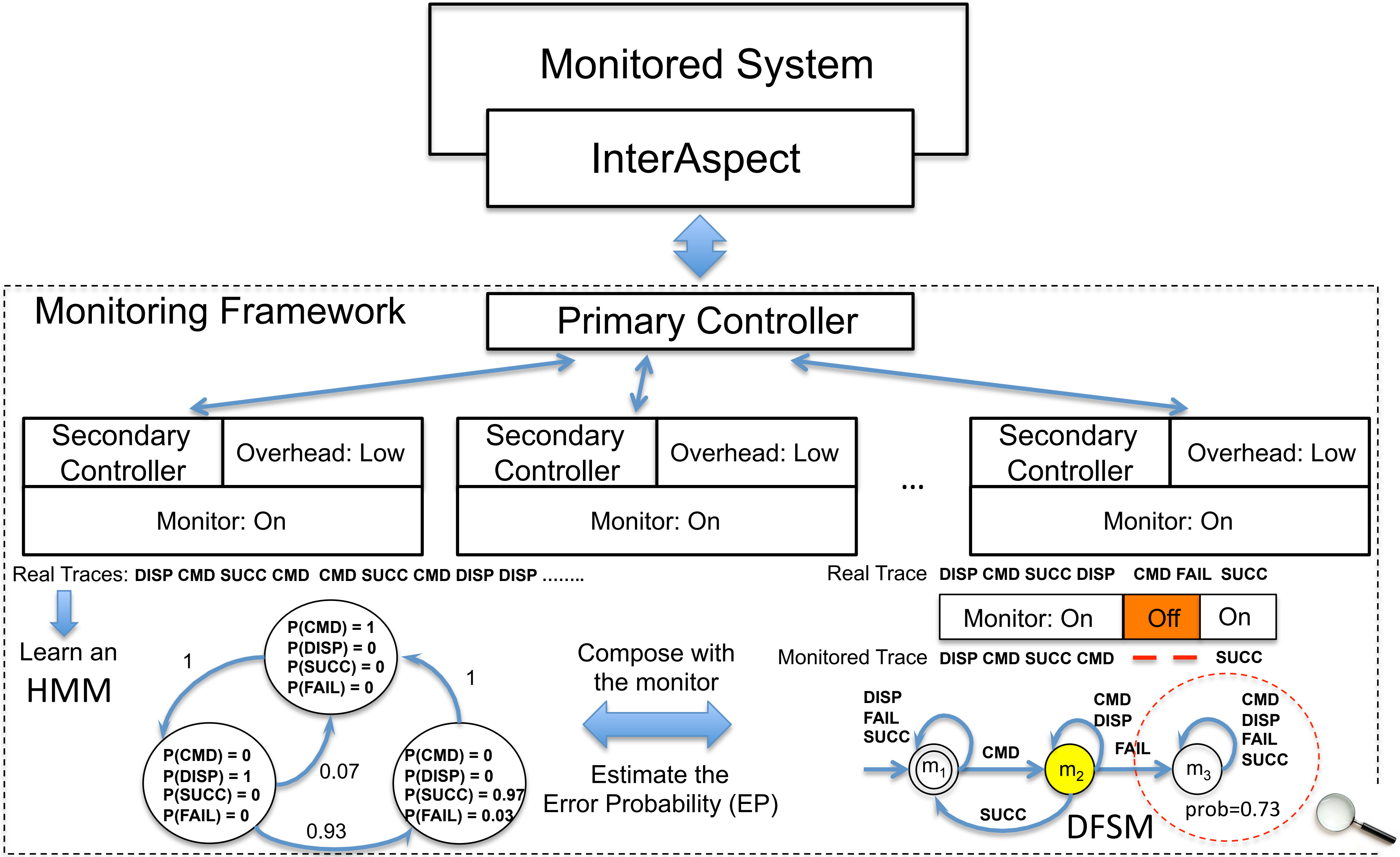


Accuracy Evaluation: Peeking



Peek after gap: **significantly improves accuracy**

Adaptive Runtime Verification Framework



Adaptive Runtime Verification Framework