

# Verification of Probabilistic Systems

Roberto Segala  
University of Verona



# Motivation

- Randomization is useful
  - Distributed protocols (consensus)
  - Symmetry breaking
- Randomization is difficult
  - Many wrong protocols
  - Intuition fails
- Main problems
  - Interplay probability/nondeterminism
  - Nondeterminism is difficult by itself



# Objectives of These Lectures

- Modeling
  - Probabilistic Automata
- Verification
  - Probabilistic progress statements
  - Coin lemmas
  - Modular verification
  - Hierarchical verification
- Examples
  - Randomized dining philosophers
  - Randomized leader election
  - Randomized consensus
  - Randomized token management



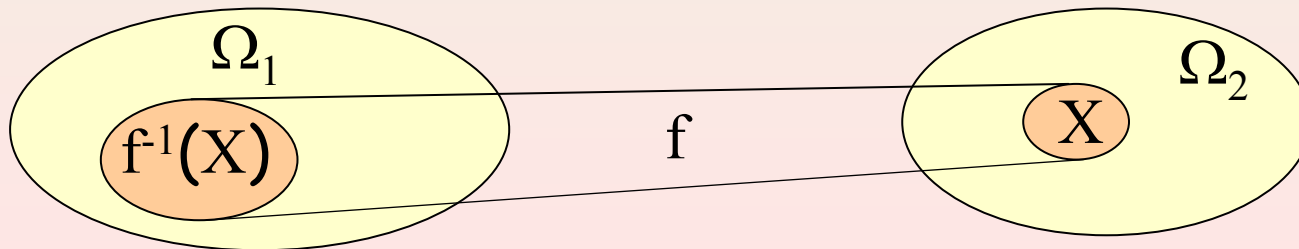
# Measure Theory

- Sample set
  - Set of objects  $\Omega$
- Sigma-field ( $\sigma$ -field)
  - Subset  $F$  of  $2^\Omega$  satisfying
    - $\sigma \in F$ 
      - Closure under complement
      - Closure under countable union
      - *Closure under countable intersection*
- Sigma-field generated by  $C \subseteq 2^\Omega$ 
  - Smallest  $\sigma$ -field that includes  $C$
- Measure on  $(\Omega, F)$ 
  - Function  $\mu$  from  $F$  to  $\mathbb{R}^{\geq 0}$  such that, for each countable collection  $\{X_i\}_I$  of pairwise disjoint sets of  $F$ ,  $\mu(\cup_I X_i) = \sum_I \mu(X_i)$
- (Sub-)probability measure
  - Measure  $\mu$  such that  $\mu(\Omega)=1$  ( $\mu(\Omega) \leq 1$ )



# Measure Theory

- Measurable function from  $(\Omega_1, \mathcal{F}_1)$  to  $(\Omega_2, \mathcal{F}_2)$ 
  - Function  $f$  from  $\Omega_1$  to  $\Omega_2$
  - For each element  $X$  of  $\mathcal{F}_2$ ,  $f^{-1}(X) \in \mathcal{F}_1$
- Image measure
  - Given a measure  $\mu$  on  $(\Omega_1, \mathcal{F}_1)$
  - $f(\mu)$  is defined on  $(\Omega_2, \mathcal{F}_2)$  by
  - $f(\mu)(X) = \mu(f^{-1}(X))$



# Probabilistic Automata

$$A = (Q, q_0, E, H, D)$$

Transition relation

$$D \subseteq Q \times (E \cup H) \times \text{Disc}(Q)$$

Internal (hidden) actions

External actions:  $E \cap H = \emptyset$

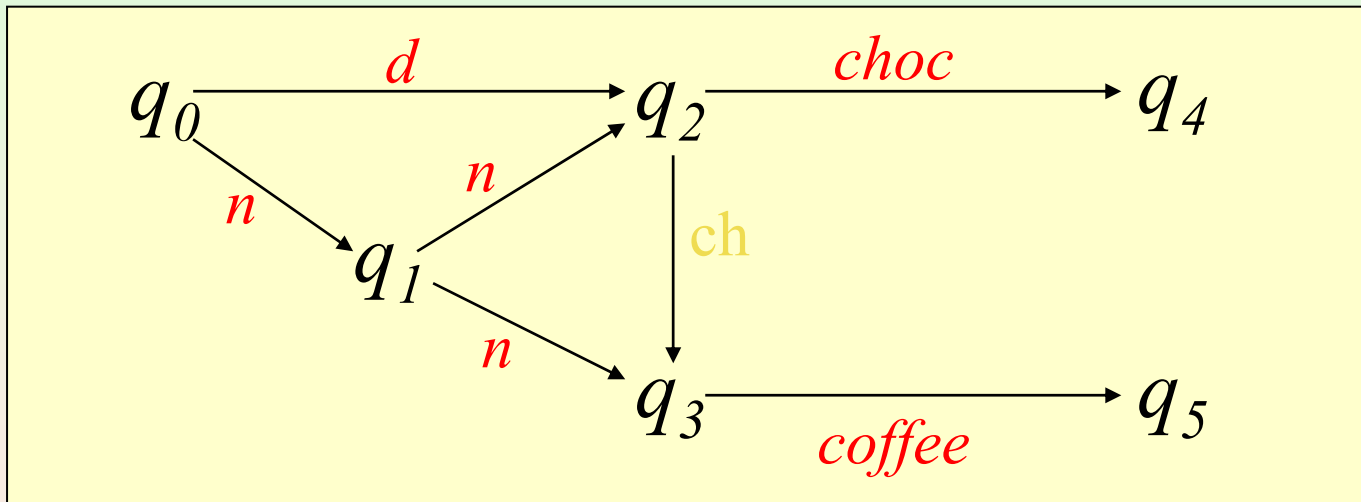
Initial state:  $q_0 \in Q$

States



# Example: Automata

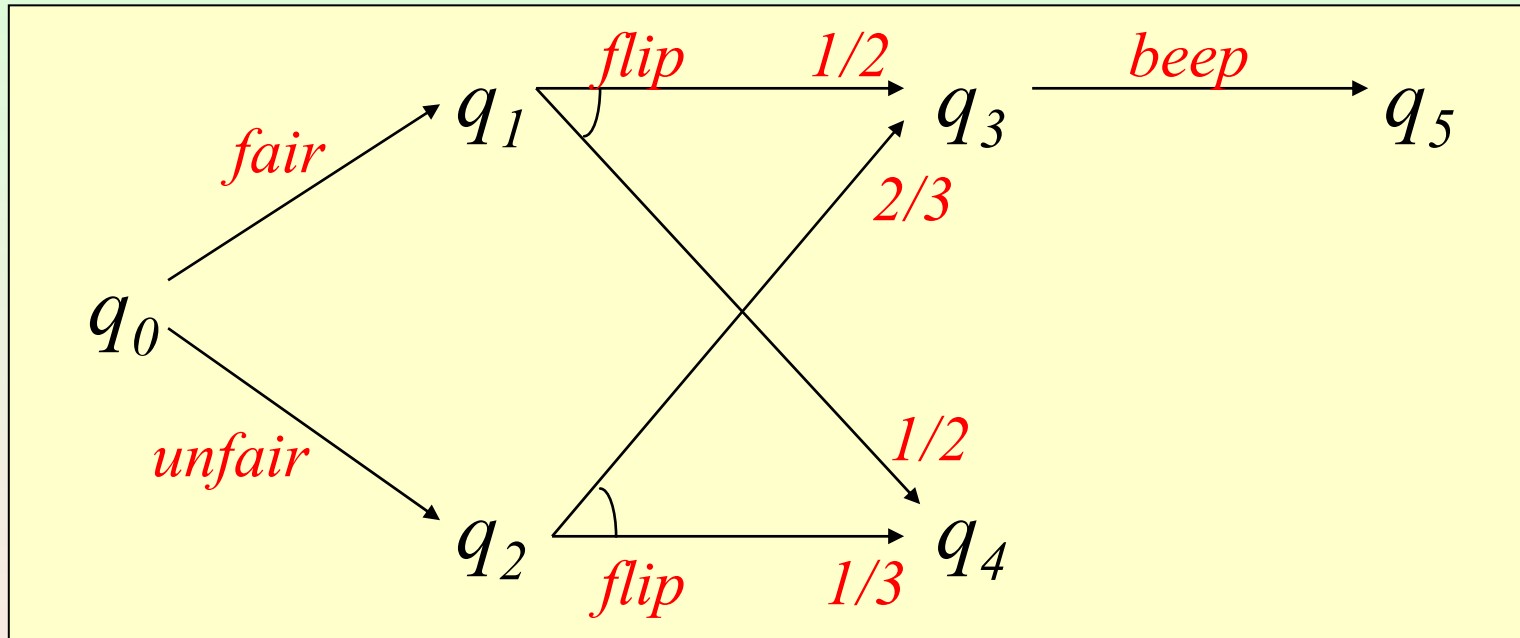
$$A = (Q, q_0, E, H, D)$$



Execution:  $q_0$   $n$   $q_1$   $n$   $q_2$   $ch$   $q_3$   $coffee$   $q_5$

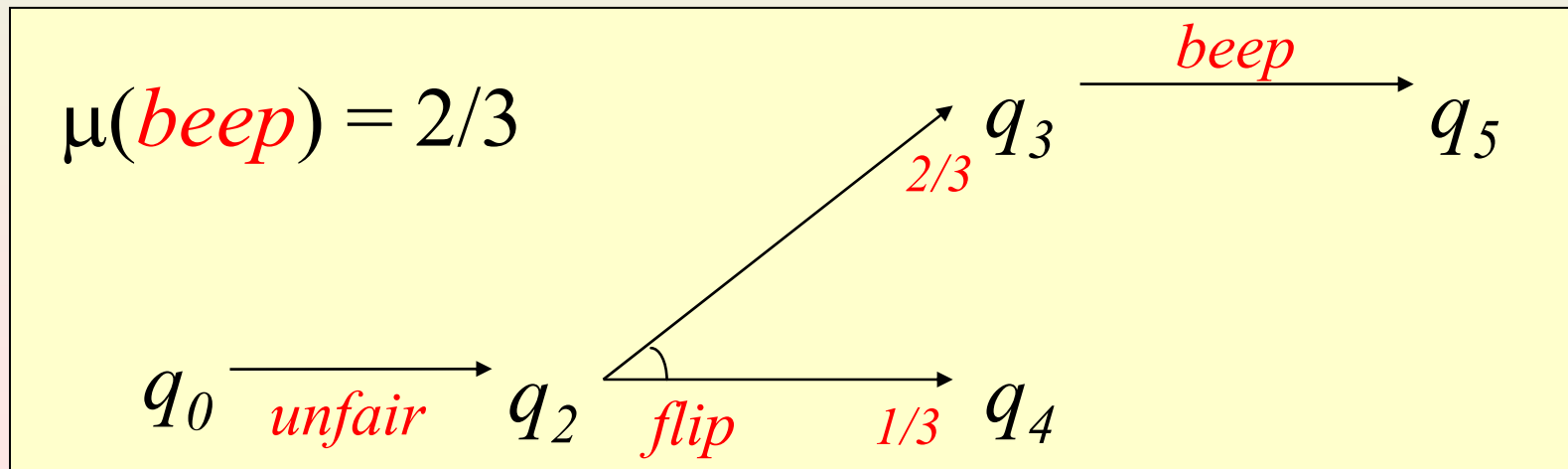
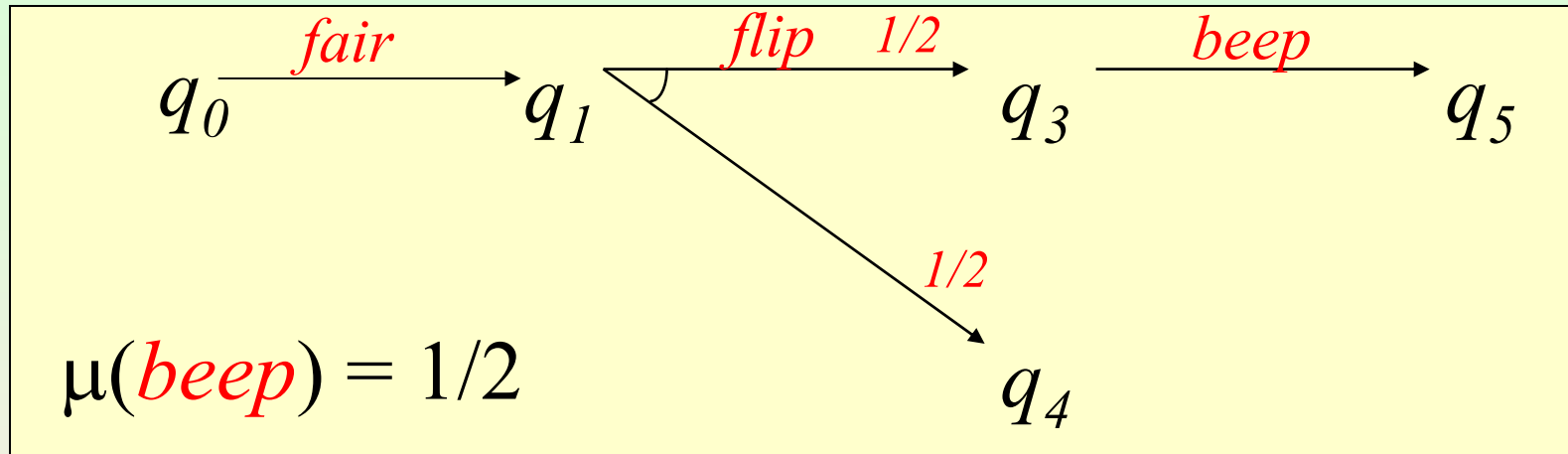
Trace:  $n$   $n$   $coffee$

# Example: Probabilistic Automata



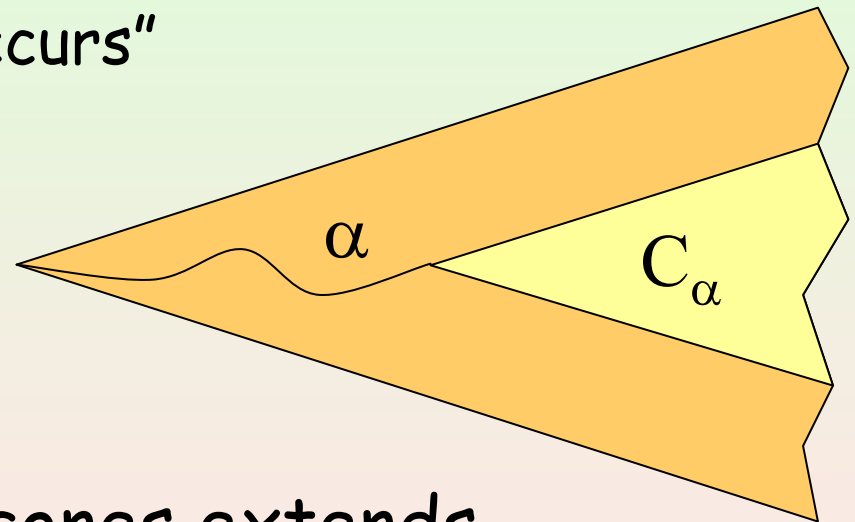
What is the probability of beeping?

# Example: Probabilistic Execution



# Cones and Measures

- Cone of  $\alpha$ 
  - Set of executions with prefix  $\alpha$
  - Represent event " $\alpha$  occurs"
- Measure of a cone
  - Product edges of  $\alpha$



Thm. The measure over cones extends uniquely to a measure over the  $\sigma$ -field generated by cones

# Schedulers and Probabilistic Executions

Scheduler  $\sigma$

$$\sigma: \text{exec}^*(A) \rightarrow \text{SubDisc}(D)$$

$$\sigma(\alpha)((q, a, \mu)) > 0 \quad \text{implies} \quad q = \text{lstate}(\alpha)$$

Probabilistic execution:

given start state  $r$ , measure  $\mu_{\sigma,r}$  where

$$\mu_{\sigma,r}(C_r) = 1$$

$$\mu_{\sigma,r}(C_{\alpha a q}) = \mu_{\sigma,r}(C_\alpha) \rho$$

$$\rho = \sum_{(s,a,v) \in D} \sigma(\alpha)((s,a,v)) v(q)$$



# Examples of Events

- Eventually action  $a$  occurs
  - Union of cones where action  $a$  occurs once
- Action  $a$  occurs at least  $n$  times
  - Union of cones where action  $a$  occurs  $n$  times
- Action  $a$  occurs at most  $n$  times
  - Complement of **action  $a$  occurs at least  $n+1$  times**
- Action  $a$  occurs exactly  $n$  times
  - Intersection of previous two events
- Action  $a$  occurs infinitely many times
  - Intersection of **action  $a$  occurs at least  $n$  times** for all  $n$
- Execution  $\alpha$  occurs and nothing is scheduled after
  - Set consisting of  $\alpha$  only
  - $C_\alpha$  intersected complement of cones that extend  $\alpha$



# Other Related Models

- Markov Decision Processes [Bel57,How60,Der70]
  - Deterministic probabilistic automata
  - A scheduler is called a policy
  - Used for optimal control
- Reactive systems [GSST90]
  - Deterministic probabilistic automata
- Labeled Concurrent Markov Chains [Var85,HJ90]
  - Probabilistic automata where each state
    - enables several labeled ordinary transitions or
    - enables a unique unlabeled transition
- Probabilistic Nondeterministic Systems [BA95]
  - Unlabeled
  - Used for model checking
- Concurrent Probabilistic Systems [BK98]
  - Unlabeled
  - Used for model checking



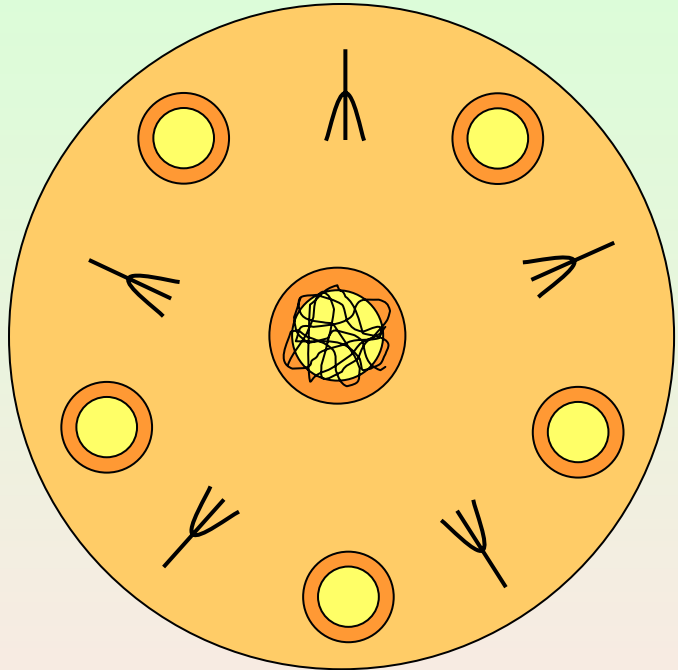
# Other Related Models

- Markov Chains
  - Unlabeled
  - No nondeterminism
  - Used for performance analysis
- Continuous time Markov Chains
  - Time is considered
  - Markov chains with exponential delays
- Generalized Stochastic Petri Nets [MBC84]
  - Petri Nets extended with exponential delays
- Stochastic Transition Systems [Alf98]
  - GSPNs with nondeterminism



# Dining Philosophers

## Algorithm of Lehmann-Rabin



Will some hungry philosopher eat eventually?

Try:

repeat

flip a coin to choose a fork  
wait to pick up the chosen fork  
if the other fork is free  
then pick it up and eat  
else put down all forks

until eating

Exit:

put down all forks

# Algorithm of Lehmann-Rabin

## Represented with Probabilistic Automata

States: for each  $i \in \{1, \dots, n\}$ ,  
 $Res_i \in \{\text{free}, \text{taken}\}$  init free  
 $u_i \in \{\text{left}, \text{right}\}$   
 $f_i, s_i \in \{1, \dots, n\}$   
 $pc_i \in \{R, F, W, S, D, P, C, EF, ES, ER\}$  init R

Actions: for each  $i \in \{1, \dots, n\}$ ,  
 $try_i, flip_i, wait_i, second_i, drop_i, crit_i,$   
 $exit_i, dropf_i, drops_i, rem_i$

$try_i$ : **pre**:  $pc_i = R$   
**post**:  $pc_i = F$

$flip_i$ : **pre**:  $pc_i = F$   
**post**:  $u_i = \text{Uniform}\{\text{left}, \text{right}\}, pc_i = W$   
 if ( $u_i = \text{left}$ )  
 then  $f_i = i-1, s_i = i$   
 else  $f_i = i, s_i = i-1$

$wait_i$ : **pre**:  $pc_i = W, Res_{f_i} = \text{free}$   
**post**:  $pc_i = S, Res_{f_i} = \text{taken}$

$second_i$ : **pre**:  $pc_i = S$   
**post**: if ( $Res_{s_i} = \text{free}$ )  
 then  $pc_i = P, Res_{s_i} = \text{taken}$   
 else  $pc_i = D$

$drop_i$ : **pre**:  $pc_i = D$   
**post**:  $pc_i = F, Res_{f_i} = \text{free}$

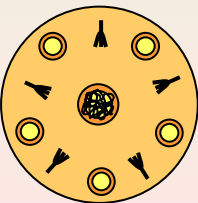
$crit_i$ : **pre**:  $pc_i = P$   
**post**:  $pc_i = C$

$exit_i$ : **pre**:  $pc_i = C$   
**post**:  $pc_i = EF$

$dropf_i$ : **pre**:  $pc_i = EF$   
**post**:  $pc_i = ES, Res_{i-1} = \text{free}$

$drops_i$ : **pre**:  $pc_i = ES$   
**post**:  $pc_i = ER, Res_i = \text{free}$

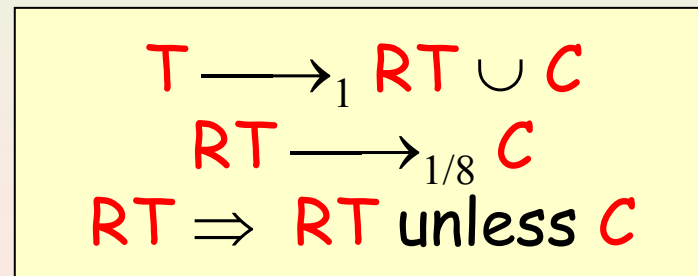
$rem_i$ : **pre**:  $pc_i = ER$   
**post**:  $pc_i = R$



# Algorithm of Lehmann-Rabin

## Progress Statements [Lynch, Saias, Segala, Pogossyants]

- T**: states where some process is trying to eat
- RT**: states of **T** where nobody is in exit
- F**: states of **RT** where somebody is ready to flip
- G**: states of **RT** where symmetry is almost broken
- P**: states of **RT** where somebody is ready to eat
- C**: states where somebody is eating



Thus, if a philosopher wants to eat, then some philosopher will eat eventually with probability 1.



# Progress Statements

$$U \xrightarrow{p} V$$

For each  $s \in U$  and each  $\sigma$

$$\mu_{\sigma,s}(\diamond V) \geq p$$

- $\diamond V$ : eventually  $V$  is reached
  - set of executions where a state from  $V$  occurs
- $\sigma$  may be restricted
  - fairness



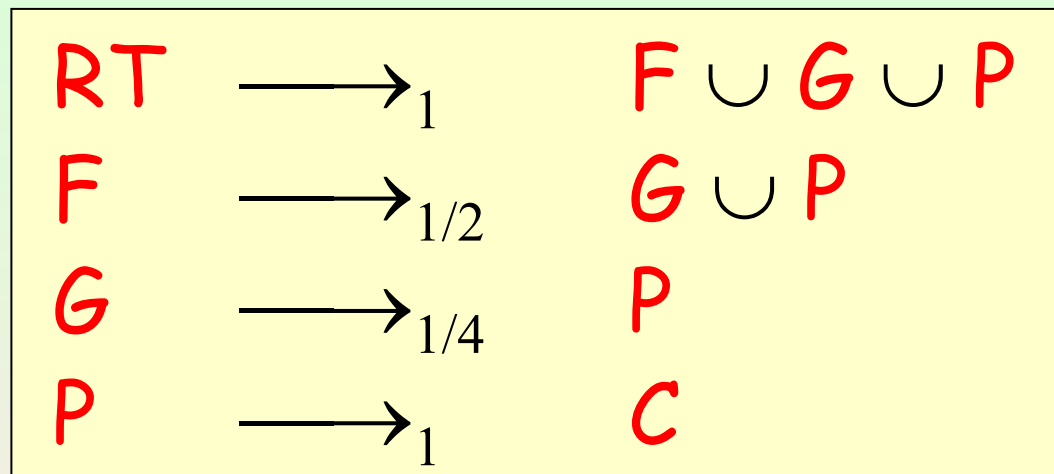
# Progress Statements: Properties

$$U \xrightarrow{p} V \text{ and } V \xrightarrow{q} X \\ \text{implies} \\ U \xrightarrow{pq} X$$

$$U \xrightarrow{p} V \text{ and } U \Rightarrow (U \text{ unless } V) \\ \text{implies} \\ U \xrightarrow{1} V$$

# Algorithm of Lehmann-Rabin

## Sequence of Statements



$RT \longrightarrow_{1/8} C$  follows by transitivity

How do we prove a statement?

# Coin Lemma

## Rule

to associate an event with each probabilistic execution

## Rule

Executions where first occurrence of  $\text{flip}_i$  leads to left or  $\text{flip}_i$  does not occur

## Lower bound

on probability of the associated event

## Lower bound

$1/2$

- We need to verify only that each event ensures progress
- Each execution where  $\text{flip}_i$  leads to left must ensures progress
- Drop right from first transition labeled by  $\text{flip}_i$
- Ordinary analysis of a nondeterministic system



# Are Coin Lemmas Useful?

## A Leader Election Protocol

- The protocol works in rounds
- At each round
  - Processes communicate and some of them play a game
  - Each player counts coin tosses until it observes head
  - There is a winner if there is a unique maximum

Protocol

Analysis

- The following property holds
  - For each  $k$ , given that there are  $k$  players, the probability of a unique maximum is at least  $\frac{1}{2}$
- Thus the protocol ends in one round with probability at least  $\frac{1}{2}$
- The protocol ends in expected two rounds



# Are Coin Lemmas Useful?

## Problems with Leader Election

- Players chosen before playing



- Players chosen while playing
  - Let two players play
  - While there is only one winner
    - Let another player play
  - Then the probability of a single winner is negligible



- Coin lemma
  - Forces us to fix  $k$  in advance
  - Event: set of executions where
    - either less than  $k$  players or
    - $k$  players and single winner



Problem  
detected



# Scheduler-Luck Games

[Dolev, Israeli, Moran]

- Two-player game
  - Scheduler: Resolves the nondeterministic choices
  - Luck: Fixes the value of some coins
- Winning strategy for Luck
  - The algorithm always completes successfully
- K-winning strategy for Luck
  - Luck wins by fixing at most k-coins

If Luck has a k-winning strategy then the algorithm completes successfully with probability at least  $1/2^k$

This is a special case of a coin lemma



# Adding Quantitative Information to Progress Statements

$$U \xrightarrow{t}_p V$$

For each  $s \in U$  and each  $\sigma$

$$\mu_{\sigma,s}(\diamond_{\leq t} V) \geq p$$

- $\diamond_{\leq t} V$ :  $V$  is reached within time  $t$ 
  - set of executions where a state from  $V$  occurs by time  $t$



# Progress Statements: Properties

$$U \xrightarrow[t1]{p} V \text{ and } V \xrightarrow[t2]{q} X$$

implies

$$U \xrightarrow[t1+t2]{pq} X$$

$$U \xrightarrow[t]{p} V \text{ and } U \Rightarrow (U \text{ unless } V)$$

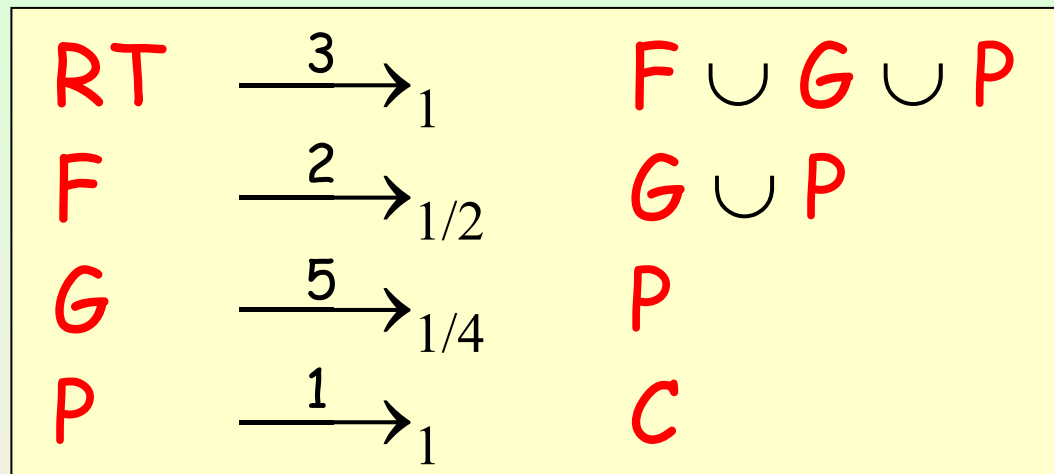
implies

expected time from  $U$  to  $V$  at most  $t/p$



# Algorithm of Lehmann-Rabin

## Adding Time



$RT \xrightarrow{11} \rightarrow_{1/8} C$  follows by transitivity

$C$  reached within expected time 88



# Composition of Probabilistic Automata

Given  $A_1 = (Q_1, q_1, E_1, H_1, D_1)$   $A_2 = (Q_2, q_2, E_2, H_2, D_2)$

define  $A_1 \parallel A_2$  to be

$$A_1 \parallel A_2 \equiv (Q_1 \times Q_2, (q_1, q_2), E_1 \cup E_2, H_1 \cup H_2, D)$$

$D = \{ (q, a, \mu_1 \times \mu_2) \text{ such that, for } i \in \{1, 2\},$

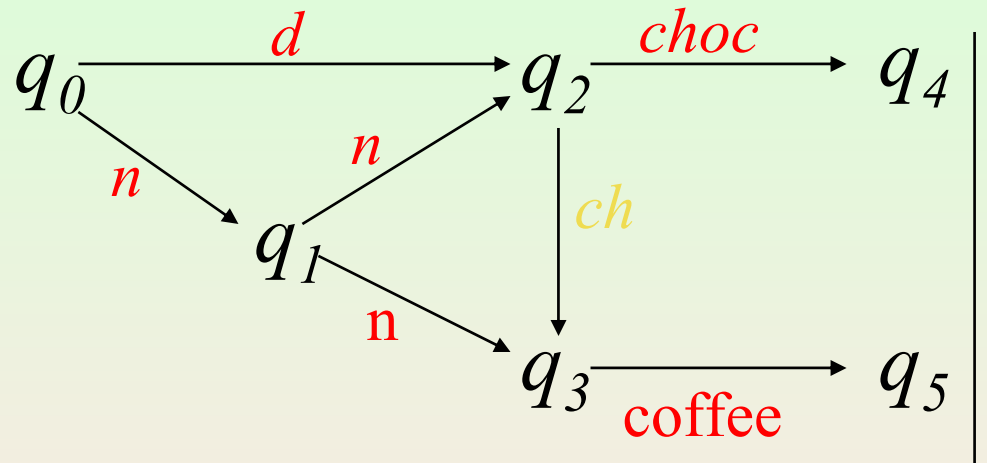
if  $a \in E_i \cup H_i$  then  $(\pi_i(q), a, \mu_i) \in D_i$

if  $a \notin E_i \cup H_i$  then  $\mu_i = \delta(\pi_i(q)) \}$

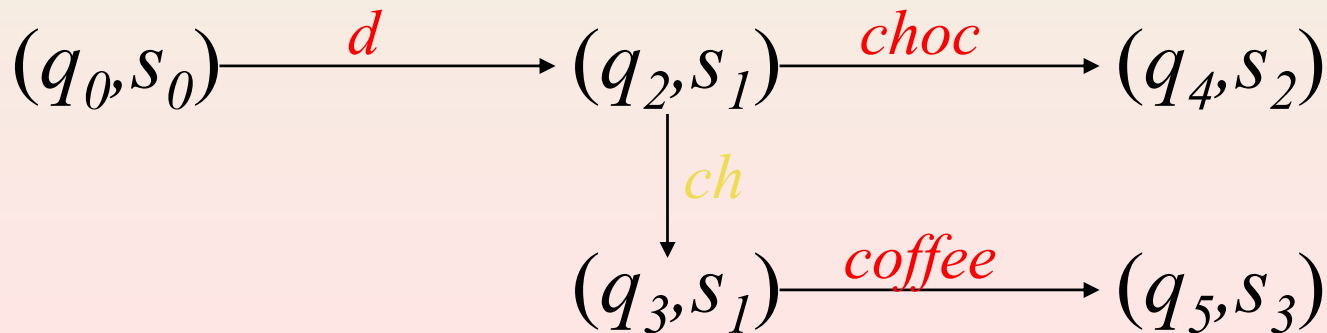
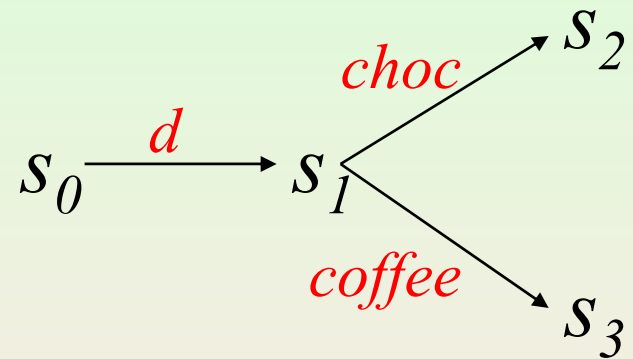


# Example: Composition

$E = \{n, d, choc, coffee\}$



$E = \{n, d, choc, coffee\}$



# Projections

Let  $\alpha$  be an execution of  $A_1 \parallel A_2$

$$\alpha = (q_0, s_0) \text{ d } (q_2, s_1) \text{ ch } (q_3, s_1) \text{ coffee } (q_5, s_3)$$

The contributions of  $A_1$  and  $A_2$  are

$$\pi_1(\alpha) \equiv q_0 \text{ d } q_2 \text{ ch } q_3 \text{ coffee } q_5$$

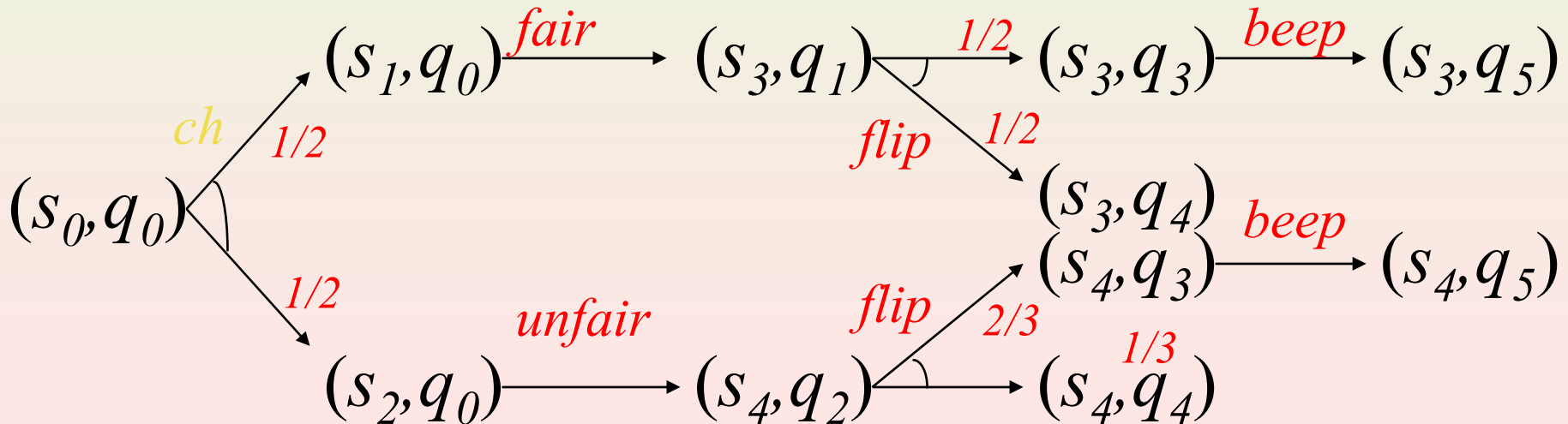
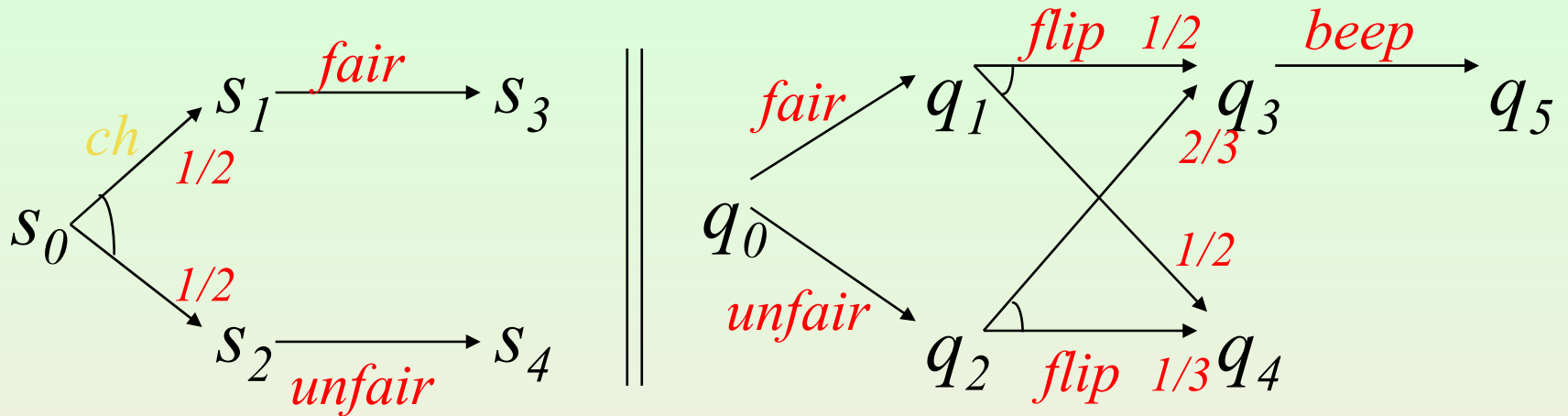
$$\pi_2(\alpha) \equiv s_0 \text{ d } s_1 \text{ coffee } s_3$$

## Theorem

$$\alpha \in \text{execs}(A_1 \parallel A_2) \quad \text{iff} \quad \forall i \in \{1, 2\} \quad \pi_i(\alpha) \in \text{execs}(A_i)$$



# Example: Composition



# Projections

The projection function is measurable

$\pi(\mu)$  : image measure under  $\pi$  of  $\mu$

## Theorem

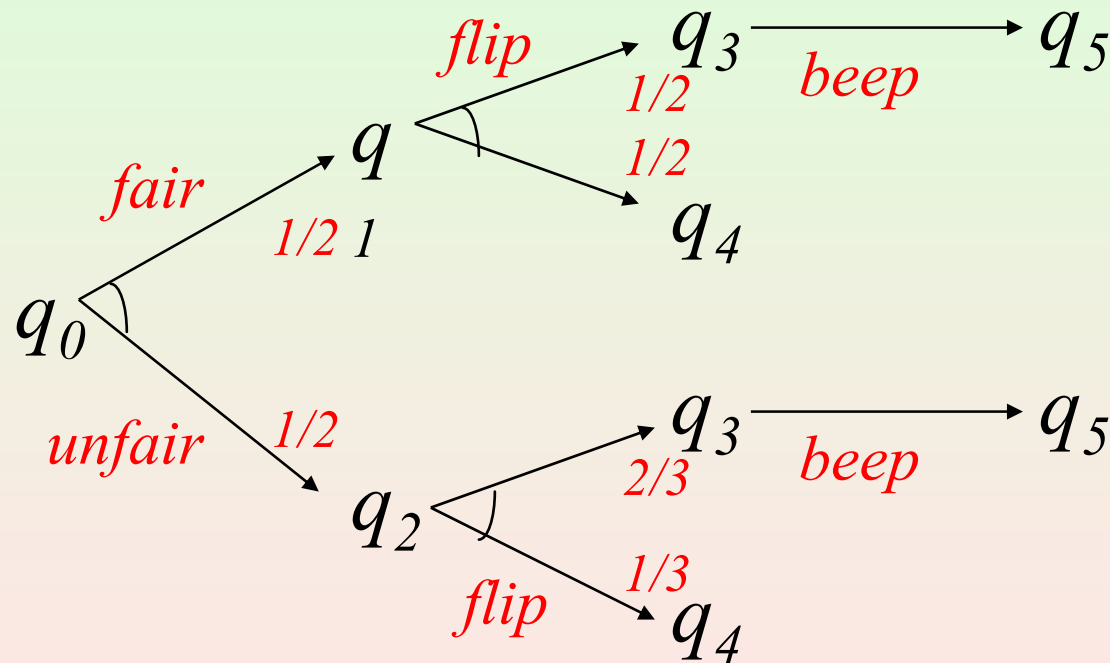
If  $\mu$  is a probabilistic execution of  $A_1 \parallel A_2$   
then

$\pi_i(\mu)$  is a probabilistic execution of  $A_i$



# Example: Projection

Projection onto right component



Note that the scheduler is randomized

# The Consensus Problem

- There are  $n$  processes
- Each process proposes a value in  $\{0,1\}$ .
- Each process may decide on a value.
- Processes may fail by stopping.

## Required properties

**Validity:** decide values that were proposed.

**Agreement:** no different decisions.

**Wait-Free Termination:** each non-failed process decides eventually.



# Randomized Consensus

**Validity:** decide values that were proposed.

**Agreement:** no different decisions.

**Probabilistic Wait-Free Termination:**  
each non-failed process decides with  
probability 1.

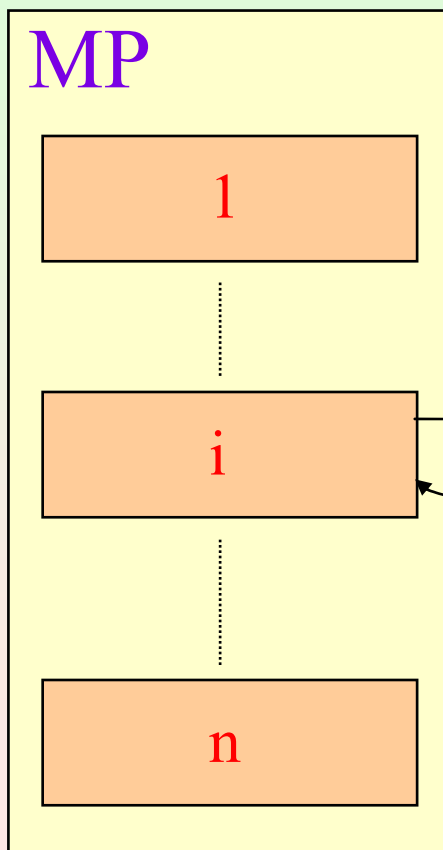
Aspnes and Herlihy

Termination within expected polynomial time.

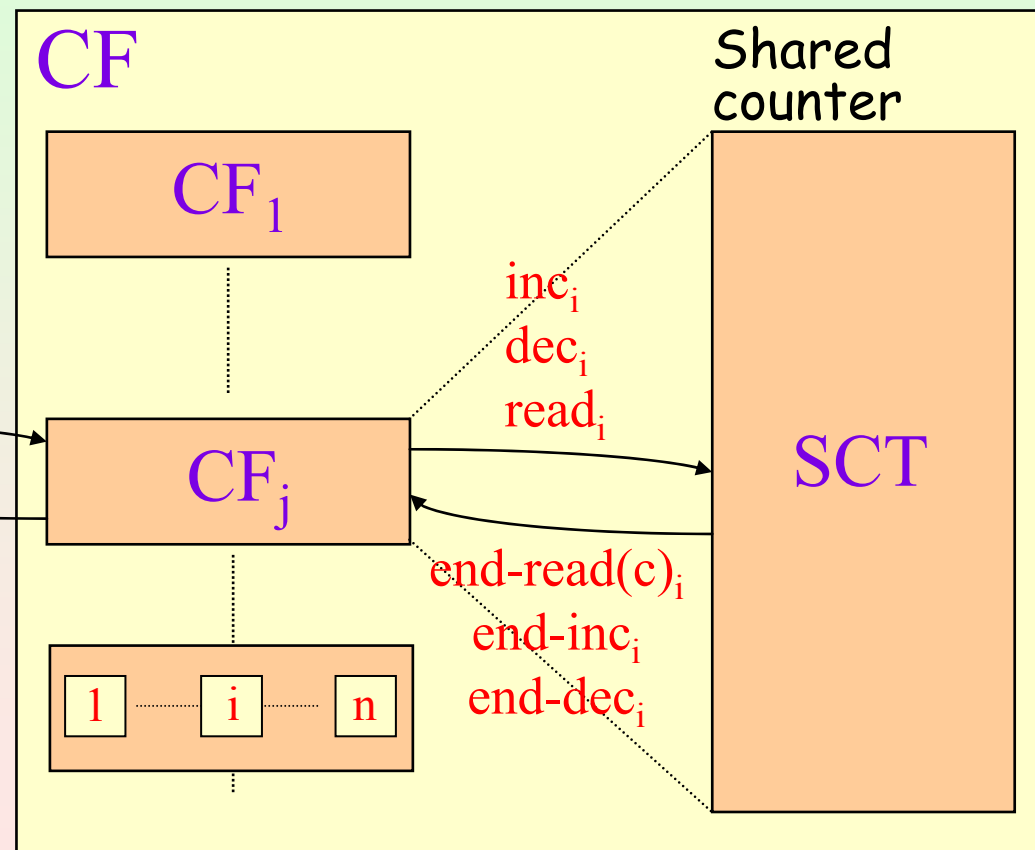


# Algorithm of Aspnes and Herlihy: Structure

Main protocol



Coin flipper



# Algorithm for the Main Protocol

```
scan pref and round
if obs-leader(i) and obs-agree(round(i)-1,v)
  then decide(v)
elseif obs-leaders-agree(v)
  then pref(i) = v and round(i) = round(i) + 1
  else pref(i) =  $\perp$ 
    scan pref and round
    if obs-leaders-agree(v)
      then pref(i) = v and round(i) = round(i) + 1
      else invoke coin flipper to choose a value for pref(i)
        round(i) = round(i) + 1
```



# Coding the Main Protocol

**Read1(k)<sub>i</sub>**

Pre:  $pc_i = \text{read1}$ ,  $k \notin \text{obs}_i$

Eff:  $\text{values}_i[k] \leftarrow \text{pref}(k)$   
 $\text{rounds}_i[k] \leftarrow \text{round}(k)$

$\text{obs}_i \leftarrow \text{obs}_i \cup \{k\}$

if  $\text{obs}_i = \{1, \dots, n\}$

then

$pc_i \leftarrow \text{check1}$

**Check2<sub>i</sub>**

Pre:  $pc_i = \text{check2}$

Eff: if  $\exists_{v \in \{0,1\}} \text{obs-leader-agree}(v)$   
then

$\text{pref}(i) \leftarrow \text{obs-leader-value}$

$\text{round}(i) \leftarrow \text{rounds}_i[i] + 1$

$\text{obs}_i \leftarrow \emptyset$

$pc_i \leftarrow \text{read1}$

else

$pc_i \leftarrow \text{flip}$



# Proof of Correctness: Safety

**Validity:** ordinary invariant proof

$\text{agree}(1,v)$  and  $\text{obs-agree}(1,v)$

**Agreement:** ordinary invariant proof

$(\text{obs-agree}(r-1,v) \text{ obs-leader}(i); \text{obs}_i = \{1,\dots,n\})$

implies

$\Rightarrow \text{agree}(r,v)$

# Invariant for Agreement

Let  $i$  be a process,  $v = \text{pref}(i)$ ,  $r = \text{round}(i)$

$\text{obs-agree}(r-1, v)_i$

$\text{fill-agree}(r, v)_i$

$\text{fill-max-round}_i = r$

implies

$\forall j : \text{obs-agree}(r, v)_j$

$\text{agree}(r, v)$

$\forall j \in \text{obs}_i : (\text{round}(j) = r-1 \text{ and } \text{pref}(j) \neq v)$

$\Rightarrow \text{fill-max-round}_j \geq r$



# Proof of Correctness: Progress

Assume the invocations to the coin flippers on non-failing ports always get an answer (M1)

$$R \xrightarrow{1} F_0 \cup F_1 \cup D$$

Assume that the all answers at round  $r$  are 0 (where, for  $s \in F_0$ ,  $\text{max-round}(s) = r$ ) (M2)

$$F_0 \xrightarrow{2} D$$



# Assumptions on Coin Flippers

Each coin flipper satisfies the properties

**C1:** each invocation on a non-failing port gets an answer with probability **1**.

**C2:** fixed a value **v** in  $\{0,1\}$  the probability that all the answers are **v** is at least **p**.

We will show that **p** is independent of **n**.



# Combination of Claims

From  $C1$  and  $C2$  and  $M1$  and  $M2$

$$\begin{array}{l} R \xrightarrow{1} F_0 \cup F_1 \cup D \\ F_0 \xrightarrow{2} D \end{array}$$

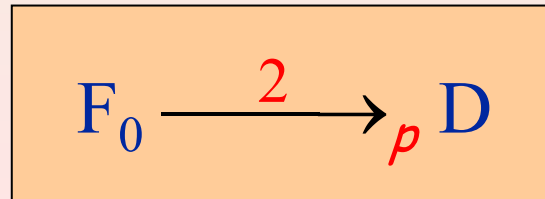
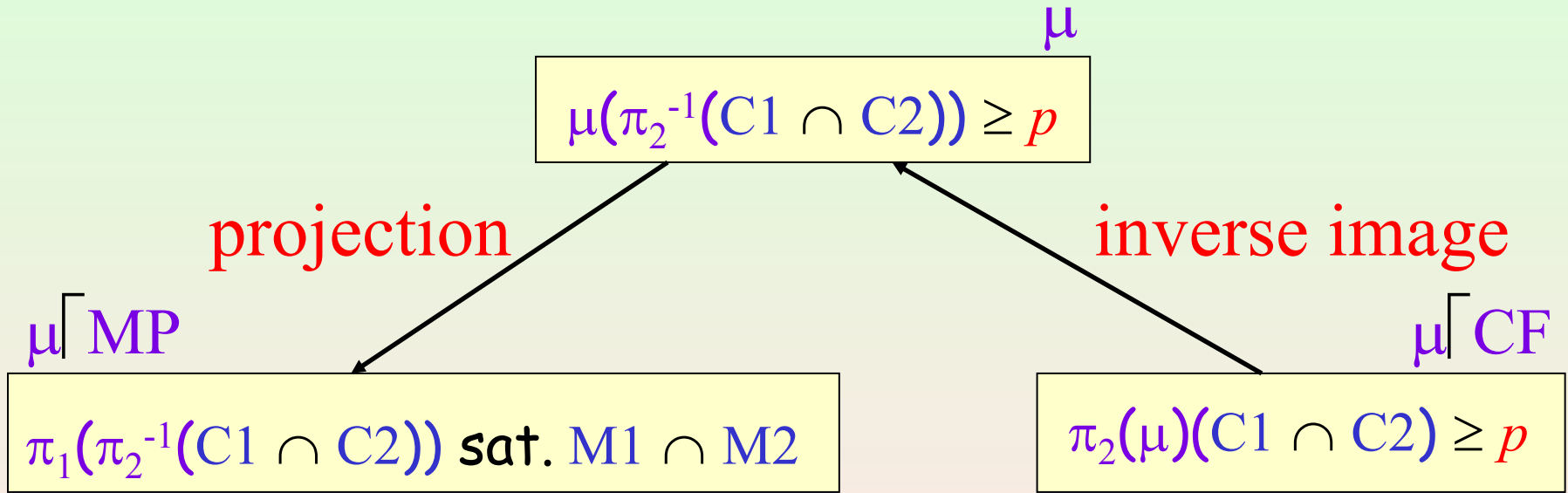
Combining the statements above

$$R \xrightarrow{3} D$$

Thus termination within expected  $3/p$  rounds.

# Formal Argument

Let  $\mu$  start in a state  $s$  of  $F_0$ .

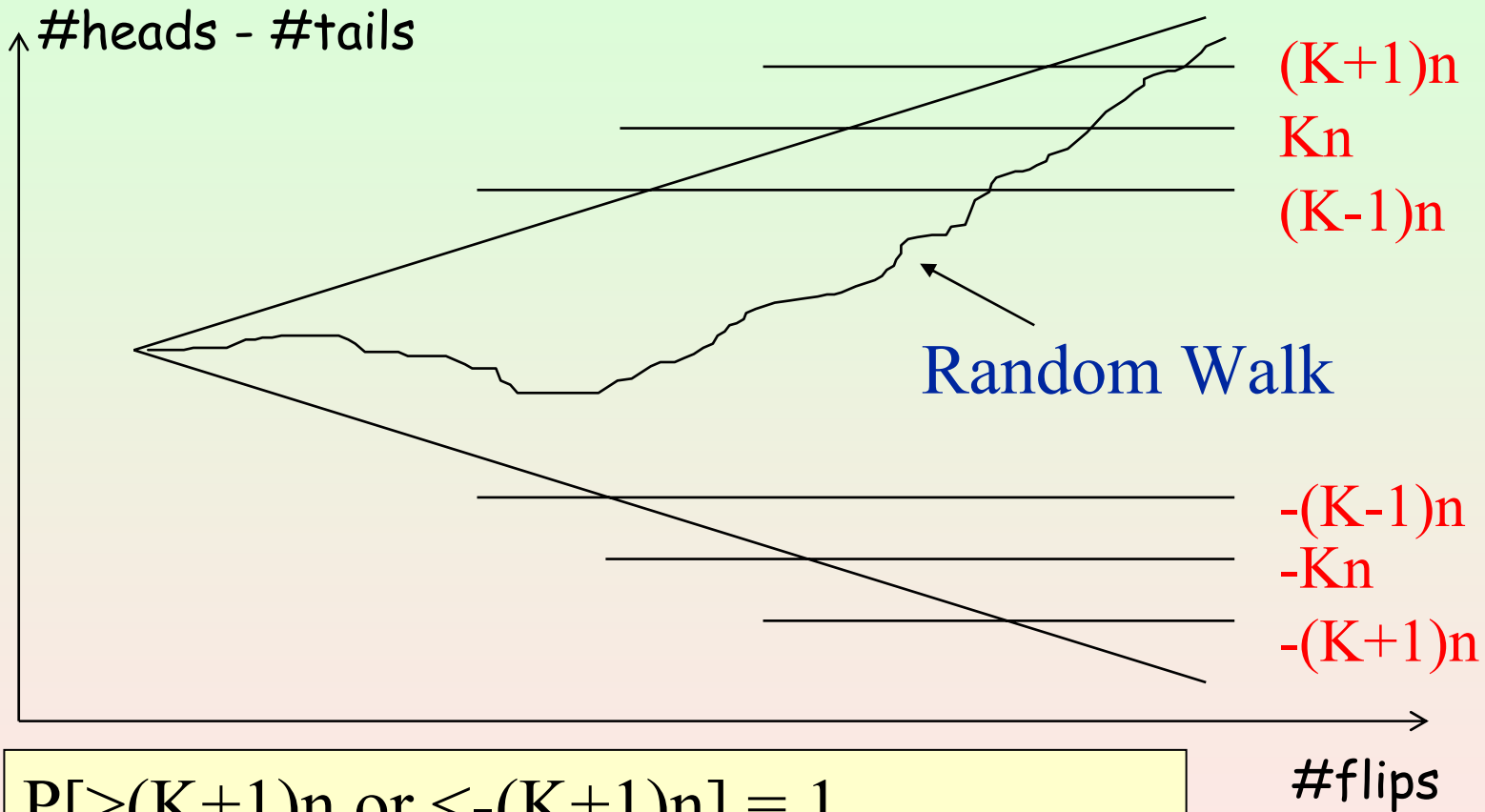


# Algorithm for $CF_{j,i}$

```
 $c = \text{FLIP}(0,1)$   
if  $c = 0$  then  $\text{inc}(\text{SCT})$   
    else  $\text{dec}(\text{SCT})$   
 $ct = \text{read}(\text{SCT})$   
if  $ct < -Kn$  then return(0)  
if  $ct > Kn$  then return(1)
```



# Why the Algorithm Works



$$P[>(K+1)n \text{ or } <-(K+1)n] = 1$$

$$P[>(K+1)n \text{ before } <-(K-1)n] = (K-1)/2K$$



# A Coin Lemma for Random Walks

Rule:

Set of executions where either  
finitely many flips, difference always  $> -(K-1)n$ , or  
difference  $\geq (K+1)n$  before  $\leq -(K-1)n$

Lower bound:

$$(K-1)/2K$$

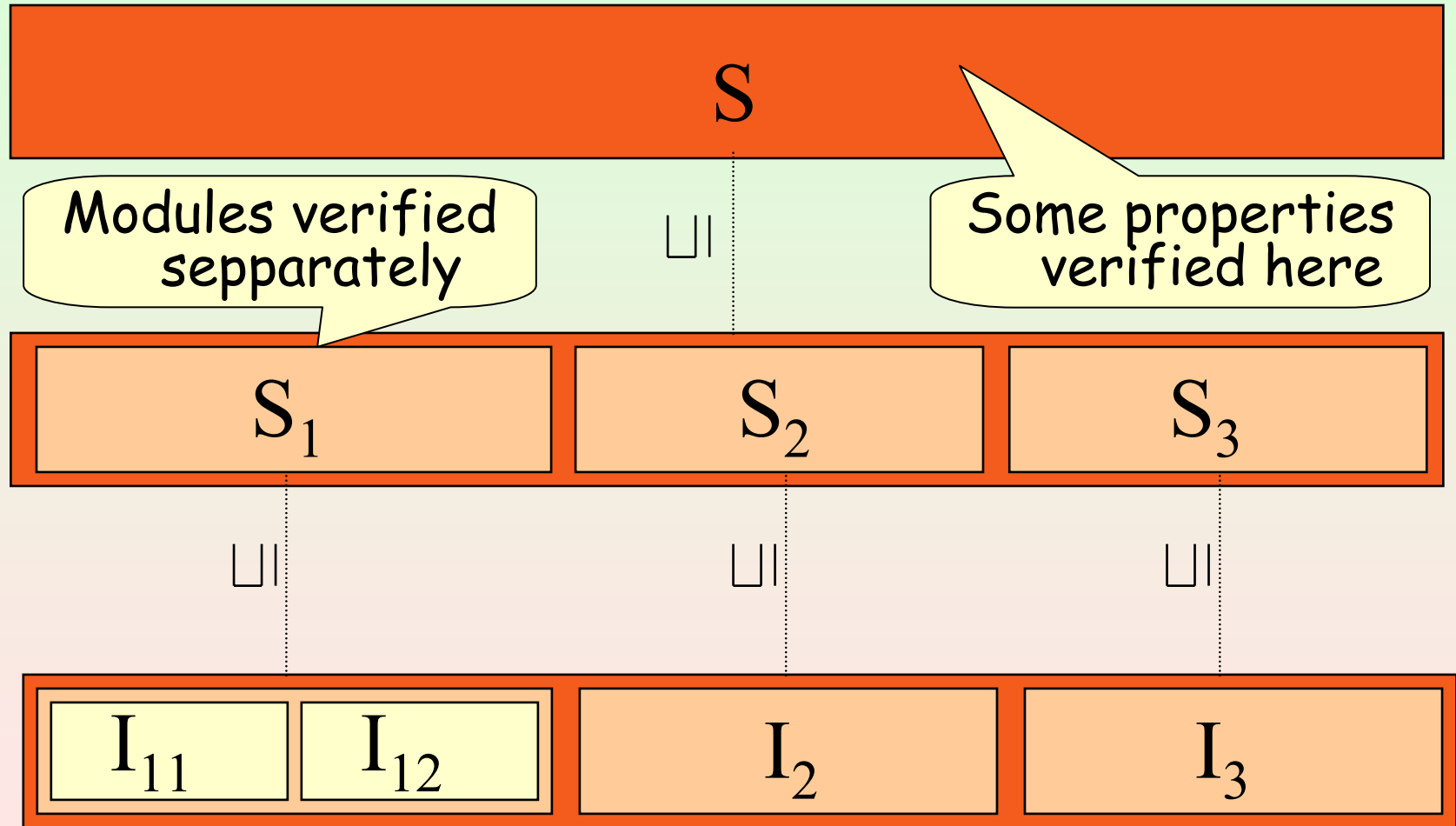


# Hierarchical Verification

- Define preorder relations on automata
  - Language inclusion
  - Simulation relations
- Use preorders to express implementation
  - A specification is an automaton
  - An implementation does only what permitted by the specification
  - The preorder preserves interesting properties
- Verify compositionally
  - Preorder preserved by composition
- Verify hierarchically
  - Refine a specification several times

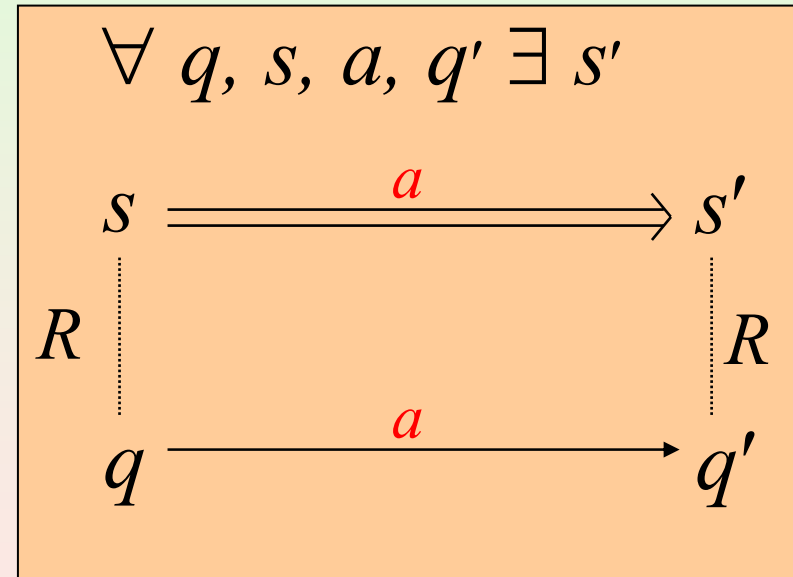
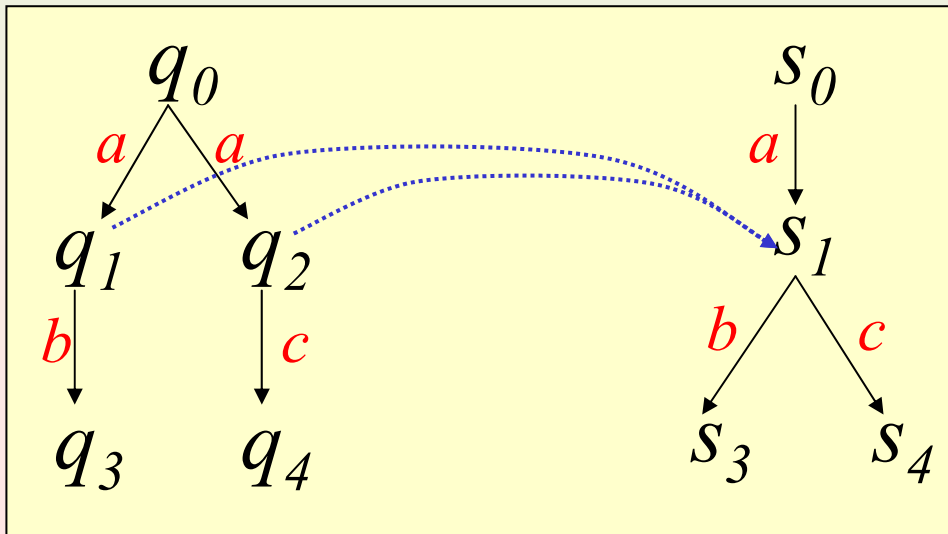


# Hyerarchical Verification: Example



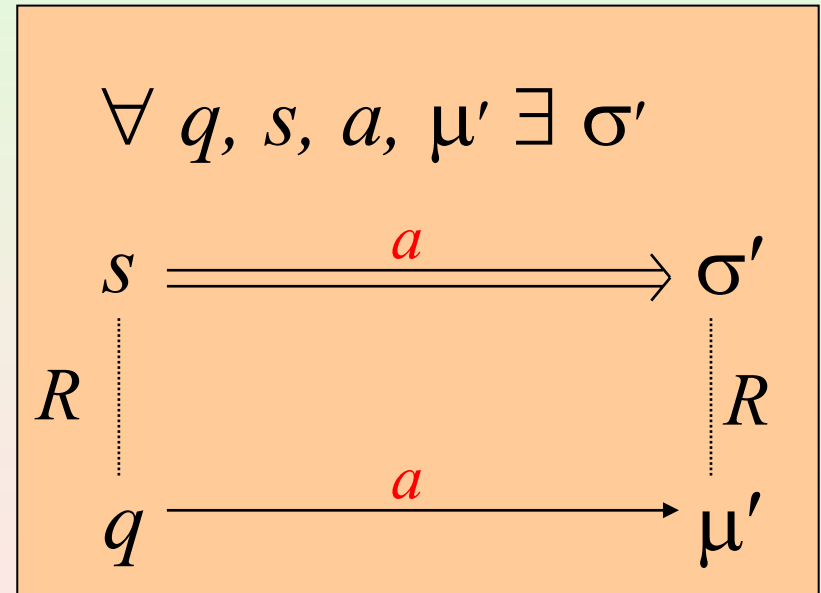
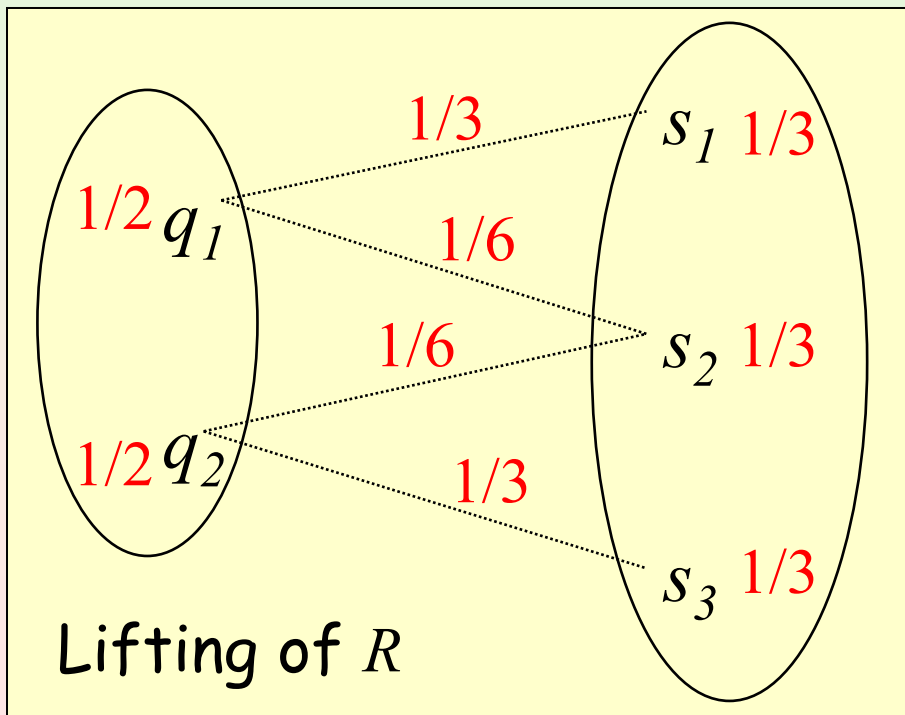
# Forward Simulations (Automata)

Forward simulation from  $A_1$  to  $A_2$  ( $A_1 \leq_F A_2$ )  
Relation  $R \subseteq Q_1 \times Q_2$  such that



# Forward Simulations

Forward simulation from  $A_1$  to  $A_2$  ( $A_1 \leq_F A_2$ )  
 Relation  $R \subseteq Q_1 \times Q_2$  such that



# Weak Transition

$$q \xRightarrow{a} \rho$$

There exists a scheduler  $\sigma$  such that

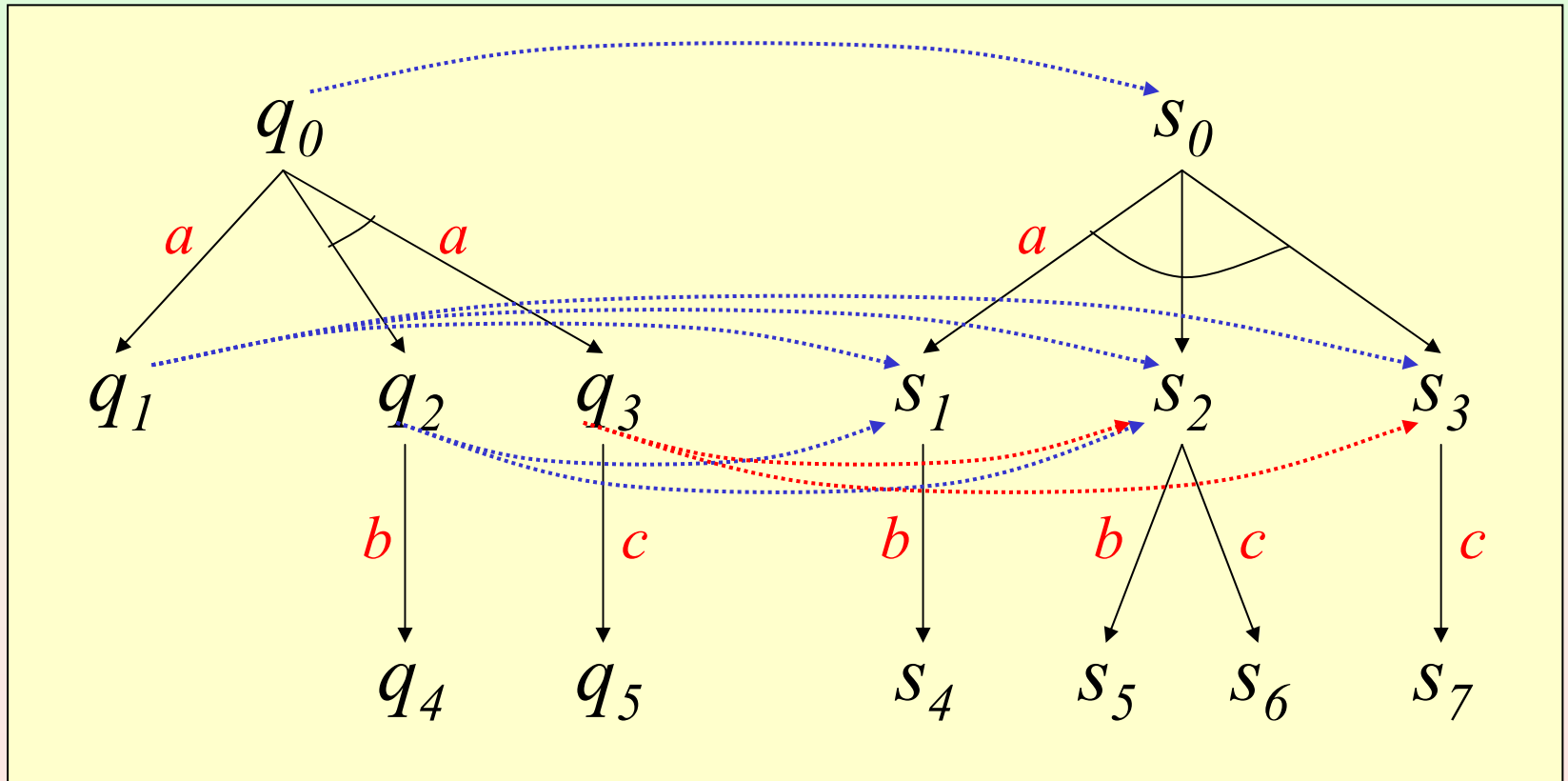
- $\mu_{\sigma,q}(exec^*) = 1$
- $\mu_{\sigma,q}(\alpha) > 0$  implies  $trace(\alpha) = a$
- $lstate(\mu_{\sigma,q}) = \rho$

where

$$lstate(\mu_{\sigma,q})(r) = \sum_{\alpha | lstate(\alpha)=r} \mu_{\sigma,q}(\alpha)$$



# Example: Simulations



# Trace Distributions

The *trace* function is measurable

Trace distribution of  $\mu$

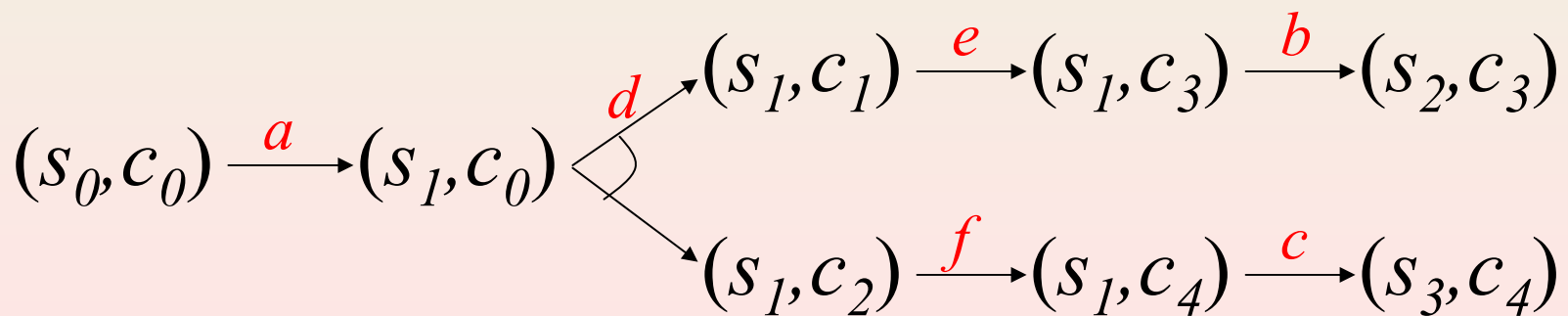
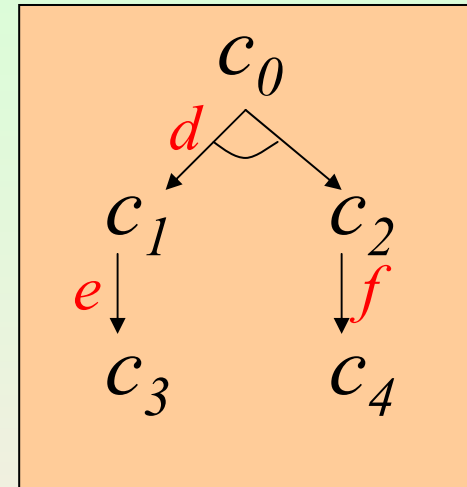
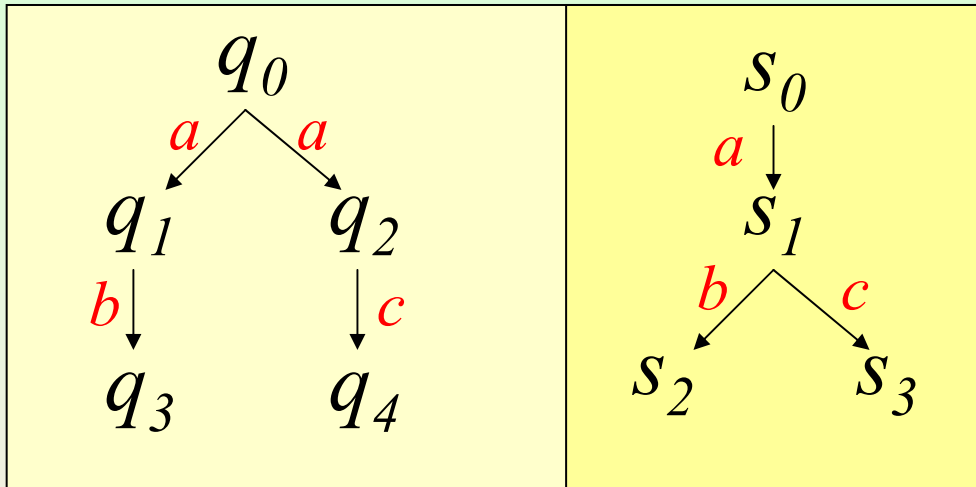
$tdist(\mu)$  : image measure under *trace* of  $\mu$

Trace distribution inclusion preorder

$A_1 \leq_{TD} A_2$  iff  $tdists(A_1) \subseteq tdists(A_2)$



# Trace Distribution Inclusion is not Compositional



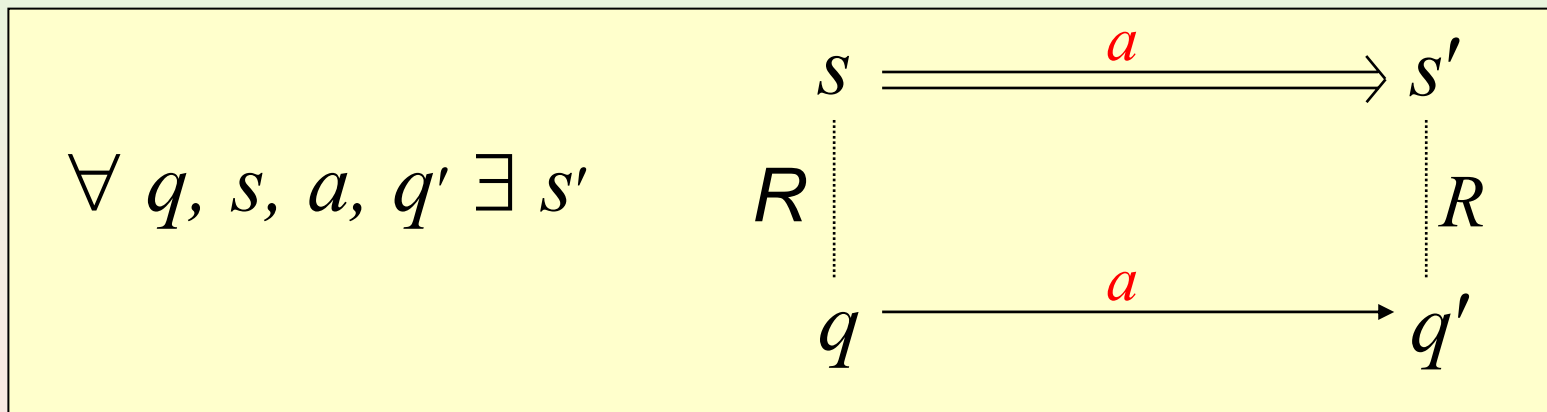
# How to Get Compositionality

- Restrict the power of composition
  - Probabilistic reactive modules [AHJ01]
  - Switched probabilistic I/O automata [CLSV04]
- Trace Distribution Precongruence
  - Coarsest precongruence included in preorder
  - Alternative characterizations
    - Principal context [Seg95]
    - Testing [Seg96]
    - Forward simulations [LSV03]



# Characterization: Forward Simulations (Automata)

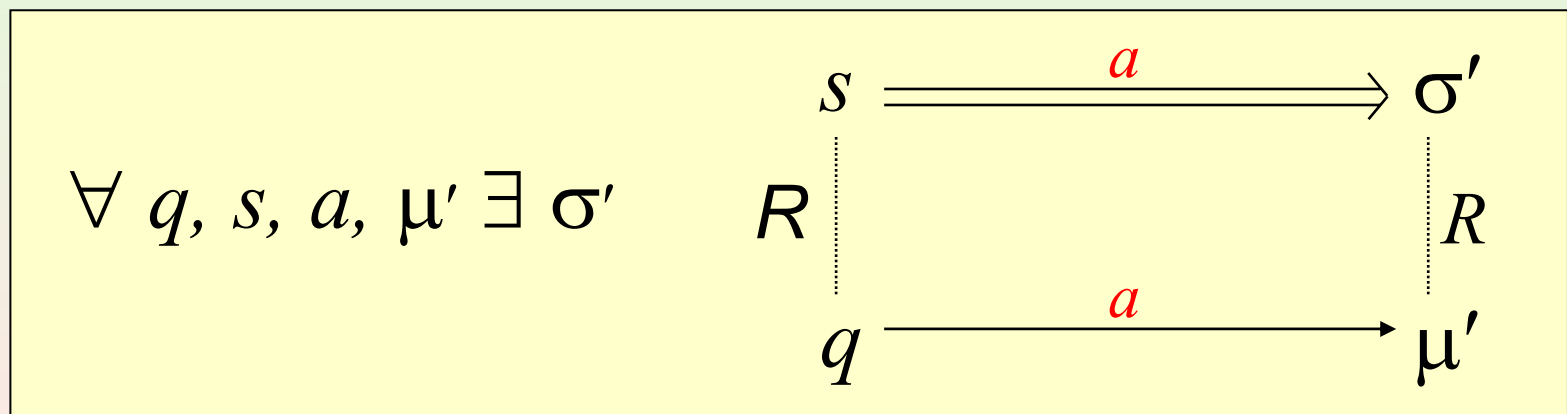
Forward simulation from  $A_1$  to  $A_2$  ( $A_1 \leq_F A_2$ )  
Relation  $R \subseteq Q_1 \times Q_2$  such that



**Theorem** [LSV02]  $A_1 \leq_F A_2$  iff  $A_1 \leq_{\text{TDC}} A_2$

# Characterization: Forward Simulations

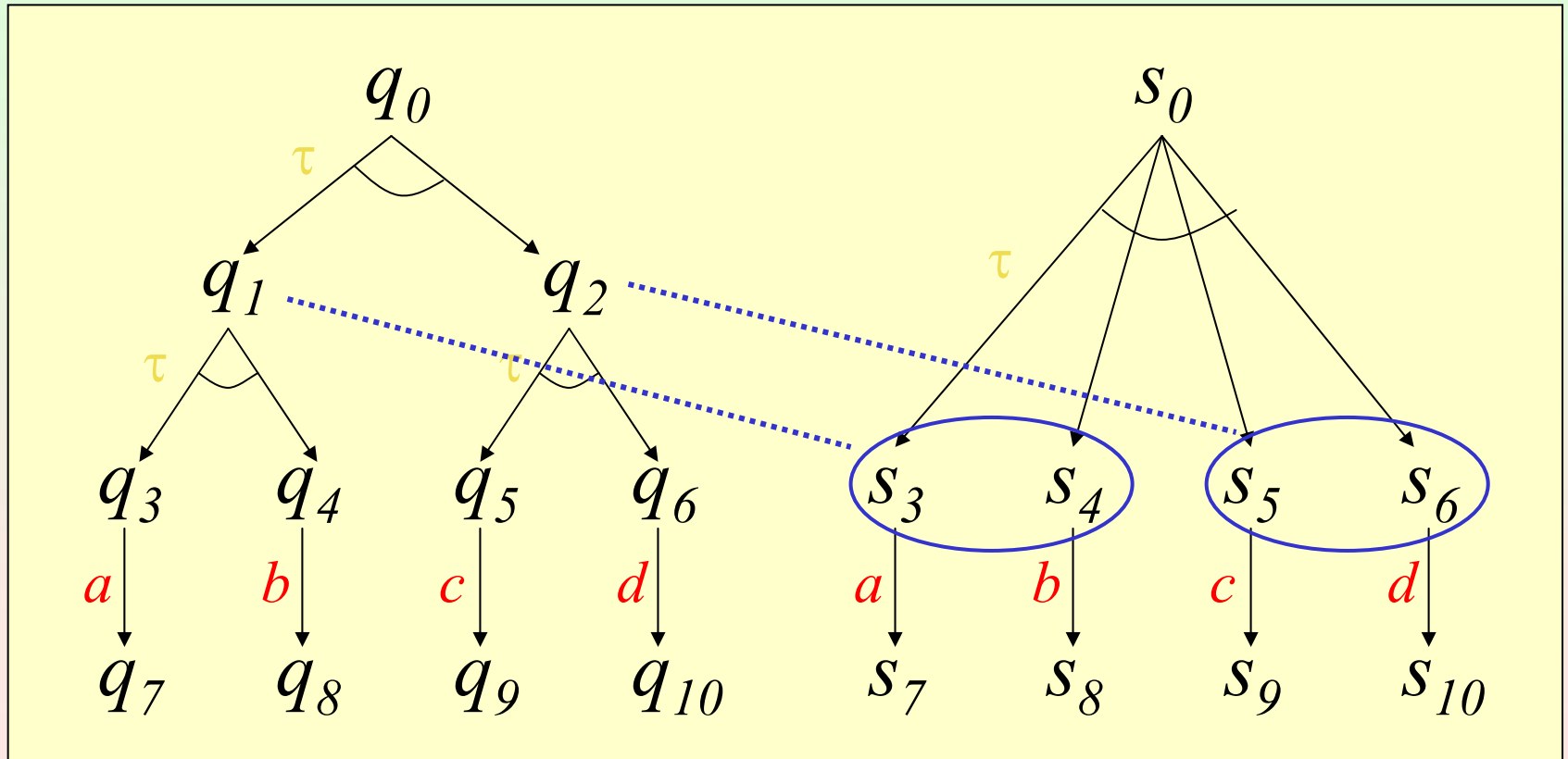
Forward simulation from  $A_1$  to  $A_2$  ( $A_1 \leq_F A_2$ )  
Relation  $R \subseteq Q_1 \times Q_2$  such that



**Theorem**  $A_1 \leq_F A_2$  implies  $A_1 \leq_{\text{TDC}} A_2$

# Example:

## Failure of Weak Forward Simulations



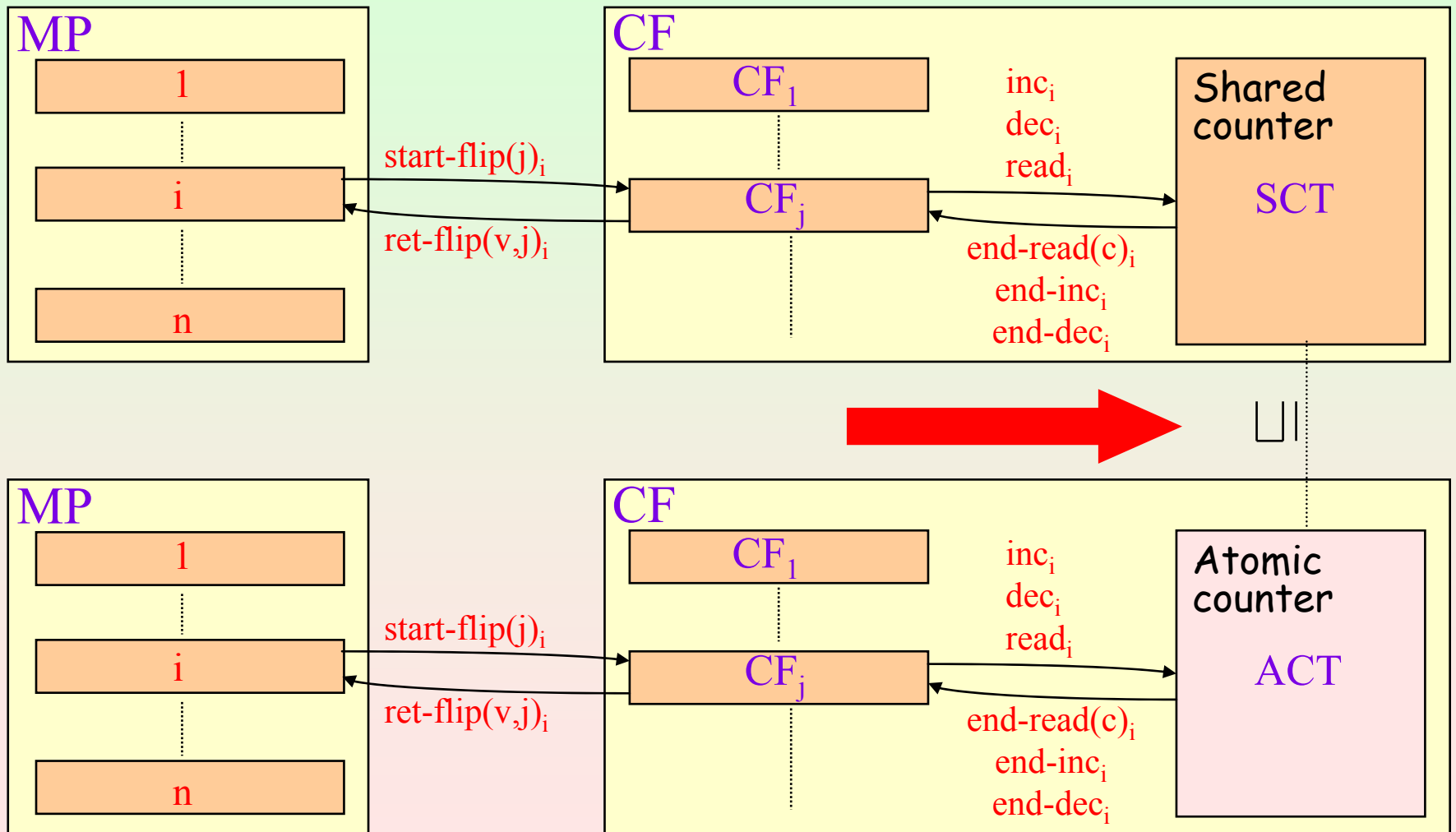
# Characterization: Probabilistic Forward Simulations

Forward simulation from  $A_1$  to  $A_2$  ( $A_1 \leq_{\text{PF}} A_2$ )  
 Relation  $R \subseteq Q_1 \times \text{Disc}(Q_2)$  such that

$$\forall q, \sigma, a, \mu' \exists \sigma'', \sigma' \quad R \begin{array}{ccc} \sigma & \xrightarrow{a} & \sigma'' \equiv \sigma' \\ \vdots & & \vdots \\ q & \xrightarrow{a} & \mu' \end{array} R$$

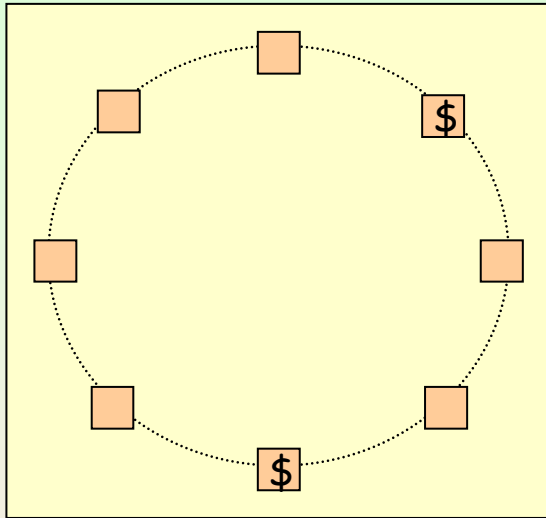
**Theorem** [LSV02]  $A_1 \leq_{\text{PF}} A_2$  iff  $A_1 \leq_{\text{TDC}} A_2$

# Back to Aspnes and Herlihy



# Self Stabilizing Tokens

[Israeli, Jalfon]



- $N$  processes in a ring topology
- Processes take turns based on tokens
- A process with the token
  - Performs its task
  - Passes the token to one of the neighbors
  - The neighbor is chosen by flipping a coin

- Problem: initially there may be more tokens
- Solution: upon collision, one token is removed
- Does the protocol stabilize?

# Analysis for Two Tokens

See [Dufлот, Fribourg, Picaronny] for general case

- Let  $d$  be the distance between the tokens
- The scheduler may choose any of the two processes holding a token
- In any case, the distance decreases with probability at least  $1/2$
- Thus in  $d$  steps, with probability at least  $1/2^d$ , there is only one token
- The distance  $d$  is at most  $m = \lceil n/2 \rceil$
- Thus, from any state, with probability at least  $1/2^m$  we reach a stable state

0/1 laws and ergodic theory for **Markov chains** ensure stability with probability 1.



# Analysis for Two Tokens Based on Progress Statements

- $U_i$ : states where tokens are at distance  $i$
- Thus,  $U_0$  is the set of stable states
- We prove

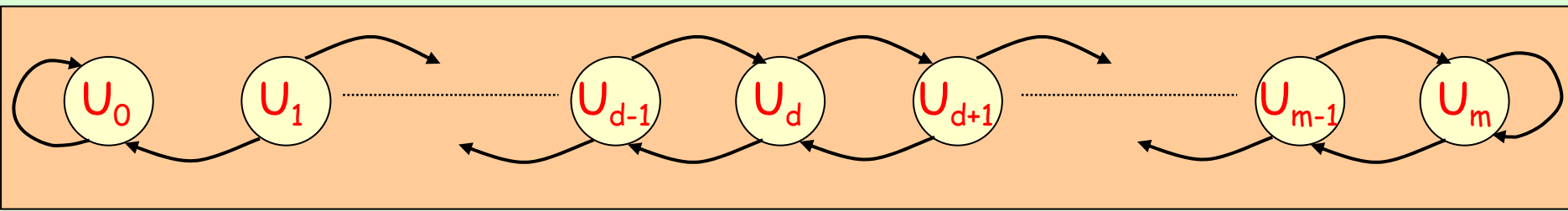
$$U_i \xrightarrow{1} \frac{1}{2} U_{i-1}$$

- By concatenation

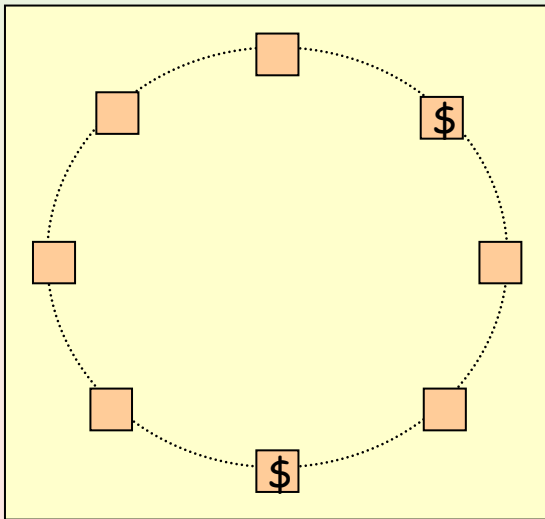
$$U \xrightarrow{m} \frac{1}{2^m} U_0$$

- Thus expected time to stability is  $m2^m$

# Analysis for Two Tokens Based on Simulation Relations



□□



- There is a simulation relation
  - States with distance  $d$  mapped to  $U_d$
- Add a *stable* action to both automata
  - Trace inclusion implies stability
- Note above process is a random walk
  - $U_0$  is an absorbing state
  - $U_m$  is a barrier
  - Stability in expected  $n^2$  steps

# Model Checking

- Describe system as an automaton
  - Most formalisms are unlabeled
  - Labels on the states
- Describe properties with temporal logic
  - LTL, CTL, PCTL, CSL, ...
- Use a model checker to verify
  - whether an automaton satisfies a formula
  - whether an automaton is a model of a formula

## Advantages

- Fully automatic
- Easy to use

## Problems

- State explosion



# Model Checkers

- **SMV, SPIN**
  - LTL and CTL model checking
  - No probability
- **Cadence SMV**
  - Model checker with enhanced proof rules
  - Combines theorem proving with model checking
- **PRISM**
  - Deals with probability and nondeterminism
- **UPPAAL**
  - Model checker for real time
  - Handles parameterized systems
  - Extensions with probability



# Verification of Probabilistic Systems with Model Checkers

- Use PRISM by itself
  - Handles only small systems
- Split problems with high level techniques
  - Verify nondeterministic part with SMV
  - Verify probabilistic part with PRISM
- Combine model checking and theorem proving
  - Theorem prover for high level proof
    - Decomposition
    - Coin lemmas
  - Model checker for low level properties
    - Prove progress statements
    - Prove events satisfy some properties

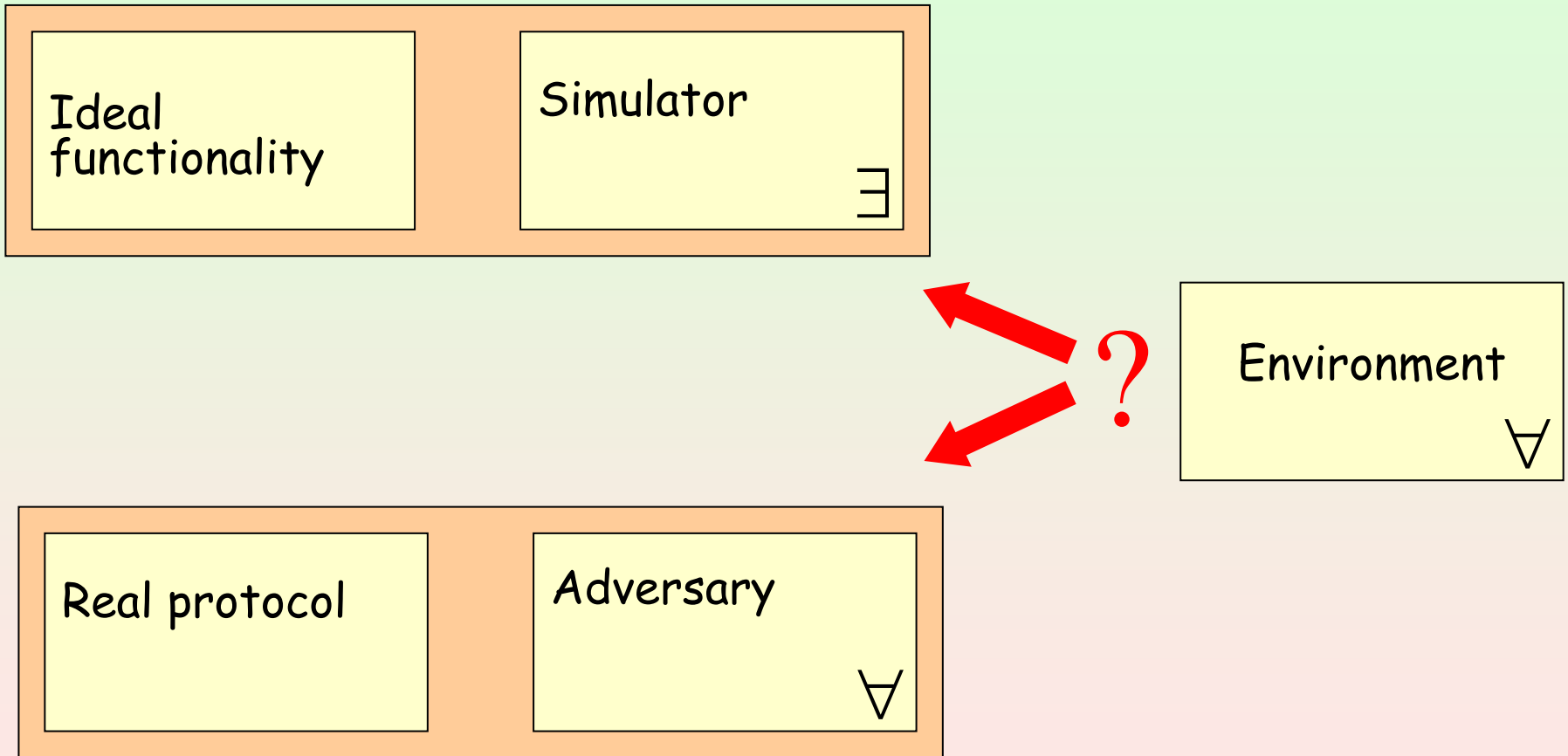


# Further Directions

- Modeling
  - Handle continuous probability measures
    - Performance analysis with nondeterminism
    - Stochastic hybrid systems
- Verification
  - Approximate simulation relations (metrics)
    - Exact values do not seem to matter
    - Useful for cryptographic protocols
  - Security protocols
    - Hierarchical analysis should work
    - Need to describe schedulers with limited power
    - Need new kinds of simulation relations



# UC-Security [Canetti]



# UC-Security with PIOAs

[Canetti, Cheung, Kaynar, Liskov, Lynch, Pereira, Segala, Vaandrager]

