

Randomized Wait-Free Consensus using An Atomicity Assumption

Ling Cheung

Institute for Computing and Information Sciences
Radboud University Nijmegen, the Netherlands

OPODIS, 12-14 December 2005, Pisa, Italy

Outline

- 1 Introduction
 - Problem Statement
 - Assumptions
- 2 Proposed Algorithm
 - Main Ideas
 - Example: Binary Consensus
 - Correctness
- 3 Model Checking with PRISM
- 4 Conclusions

Randomized Consensus

A protocol run by N parallel processes, each given an initial preference value.

Goal: To agree on a single preference value.

Randomized Consensus

A protocol run by N parallel processes, each given an initial preference value.

Goal: To agree on a single preference value.

Correctness Criteria:

- (*Validity*) Final decision value was the initial preference of some process.
- (*Agreement*) No two processes decide on different values.
- (*Termination*) Every live process eventually decides on some value.

Randomized Consensus

A protocol run by N parallel processes, each given an initial preference value.

Goal: To agree on a single preference value.

Correctness Criteria:

- (*Validity*) Final decision value was the initial preference of some process.
- (*Agreement*) No two processes decide on different values.
- (*Termination*) Every live process eventually decides on some value.

Randomization: processes can toss coins.

Randomized Consensus

A protocol run by N parallel processes, each given an initial preference value.

Goal: To agree on a single preference value.

Correctness Criteria:

- (*Validity*) Final decision value was the initial preference of some process.
- (*Agreement*) No two processes decide on different values.
- (*Termination*) Every live process eventually decides on some value.

Randomization: processes can toss coins.

- (*Probabilistic Termination*) With probability 1, every live process eventually decides on some value.

Assumptions

- **Failure model:**
 - undetected, irreversible crash failures;
 - up to $N - 1$ failures (i.e., *wait-free*).

Assumptions

- **Failure model:**
 - undetected, irreversible crash failures;
 - up to $N - 1$ failures (i.e., *wait-free*).
- **Communication:**
 - asynchronous communication via shared memory;
 - *multi-writer multi-reader* (MWMR) atomic registers.

Assumptions

- **Failure model:**
 - undetected, irreversible crash failures;
 - up to $N - 1$ failures (i.e., *wait-free*).
- **Communication:**
 - asynchronous communication via shared memory;
 - *multi-writer multi-reader* (MWMR) atomic registers.
- **Complexity measure:**
 - expected total number of Rd/Wrt memory operations.

Assumptions

- **Failure model:**
 - undetected, irreversible crash failures;
 - up to $N - 1$ failures (i.e., *wait-free*).
- **Communication:**
 - asynchronous communication via shared memory;
 - *multi-writer multi-reader* (MWMR) atomic registers.
- **Complexity measure:**
 - expected total number of Rd/Wrt memory operations.
- **Adversary model:**
 - atomic random-write operation.

Adversary Models

Given finite history, **adversaries** determine which process performs the next operation.

Adversary Models

Given finite history, **adversaries** determine which process performs the next operation.

Their “goal” is to prevent consensus: model adverse conditions in computation environment.

Worst expected complexity: under worst possible adversary.

Adversary Models

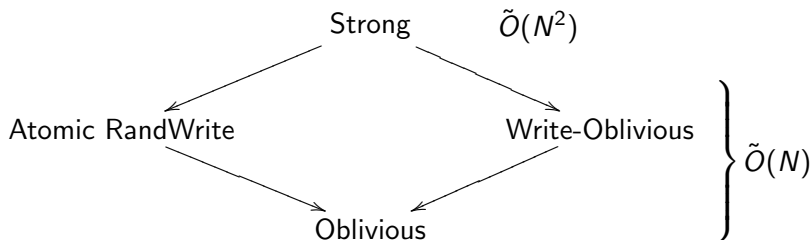
Given finite history, **adversaries** determine which process performs the next operation.

Their “goal” is to prevent consensus: model adverse conditions in computation environment.

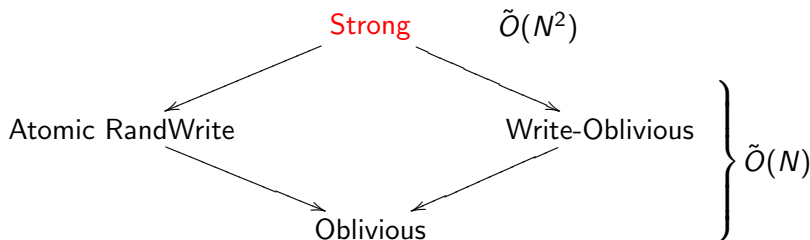
Worst expected complexity: under worst possible adversary.

Adversaries may have **complete** or **partial** access to dynamic information, thus different complexity results.

Adversary Models and Expected Total Work



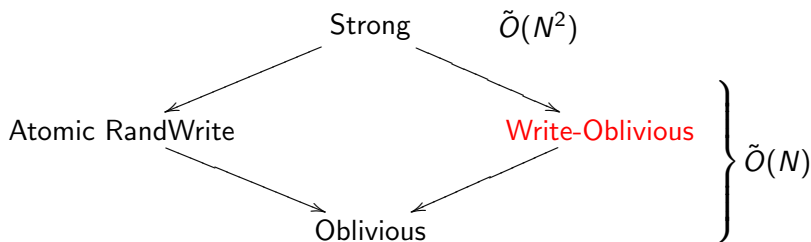
Adversary Models and Expected Total Work



Complete information over execution history.

- Bracha and Rachman, 1991: $O(N^2 \log N)$
- Aspnes, 1998: $\Omega(N^2 / \log^2 N)$

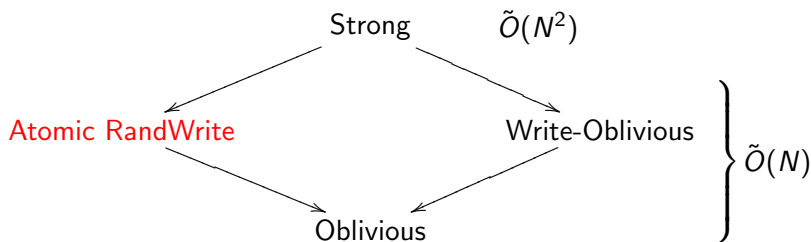
Adversary Models and Expected Total Work



Consensus gets easier when adversaries “know” less.

Example: $O(N \log N)$ against *write-oblivious* adversaries in MWMR model [Aumann, 1997].

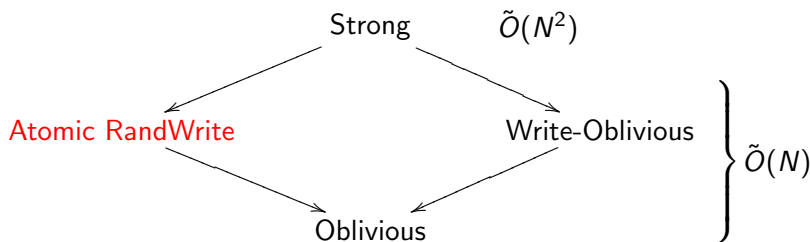
Adversary Models and Expected Total Work



This paper: coin flip and write in one atomic step.

Expected total work $O(N \log(\log N))$.

Adversary Models and Expected Total Work



This paper: coin flip and write in one atomic step.

Expected total work $O(N \log(\log N))$.

Based on [Chor, Isreali and Li, 1994]: $O(N^2)$.

Proposed Algorithm: Main Ideas

Consensus as a **race amongst preference values**, using a round structure.

Proposed Algorithm: Main Ideas

Consensus as a **race amongst preference values**, using a round structure.

- Each process “supports” one value: advance with prob. $\frac{1}{2N}$.

Proposed Algorithm: Main Ideas

Consensus as a **race amongst preference values**, using a round structure.

- Each process “supports” one value: advance with prob. $\frac{1}{2N}$.
- **Breaking symmetry**:
 - “lucky” values move ahead, “unlucky” ones stay put;

Proposed Algorithm: Main Ideas

Consensus as a **race amongst preference values**, using a round structure.

- Each process “supports” one value: advance with prob. $\frac{1}{2N}$.
- **Breaking symmetry**:
 - “lucky” values move ahead, “unlucky” ones stay put;
 - processes switch from slow values to fast ones.

Proposed Algorithm: Main Ideas

Consensus as a **race amongst preference values**, using a round structure.

- Each process “supports” one value: advance with prob. $\frac{1}{2N}$.
- **Breaking symmetry**:
 - “lucky” values move ahead, “unlucky” ones stay put;
 - processes switch from slow values to fast ones.
- Two-round lead guarantees consensus.

Proposed Algorithm: Main Ideas

Consensus as a **race amongst preference values**, using a round structure.

- Each process “supports” one value: advance with prob. $\frac{1}{2N}$.
- **Breaking symmetry**:
 - “lucky” values move ahead, “unlucky” ones stay put;
 - processes switch from slow values to fast ones.
- Two-round lead guarantees consensus.

[Chor, Israeli and Li, 1994]: **race amongst processes** in SWMR model.

Example: Binary Consensus with 4 Processes

$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

Claim: with probability $1 - \frac{1}{N}$,
protocol finishes before the 1's reach
the top.

v_0	v_1
0	0
0	0
0	0
0	0
0	0
1	1

Example: Binary Consensus with 4 Processes

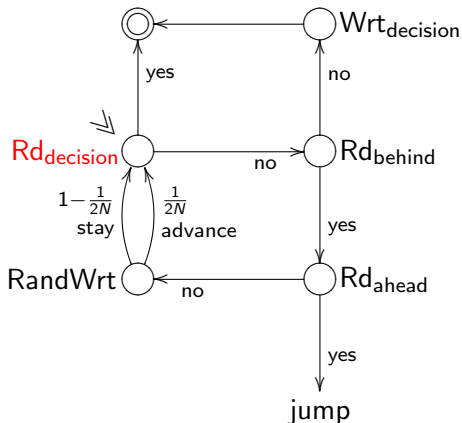
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
<u>0</u>	0
0	0
0	0
0	0
0	0
1	1



Example: Binary Consensus with 4 Processes

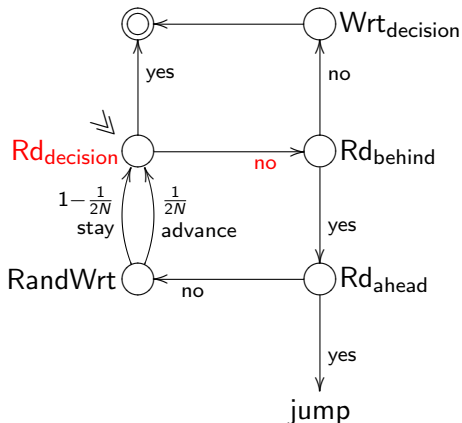
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	<u>0</u>
0	0
0	0
0	0
0	0
1	1



Example: Binary Consensus with 4 Processes

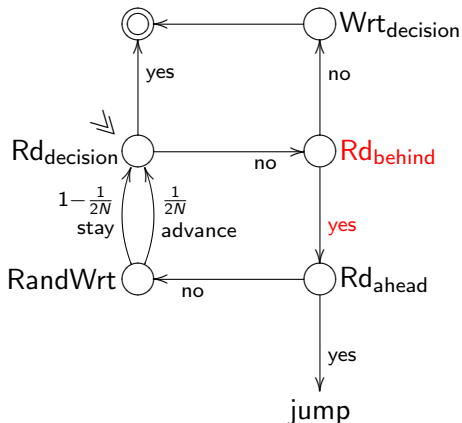
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	0
0	0
1	1



Example: Binary Consensus with 4 Processes

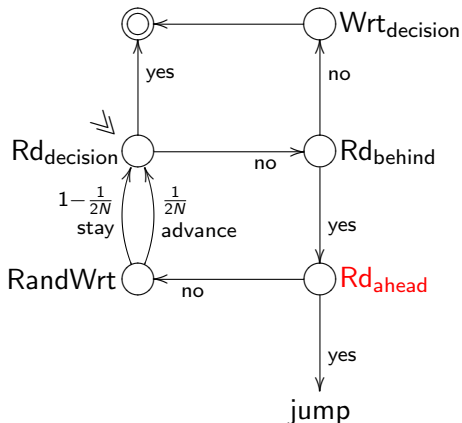
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	0
<u>0</u>	0
1	1



Example: Binary Consensus with 4 Processes

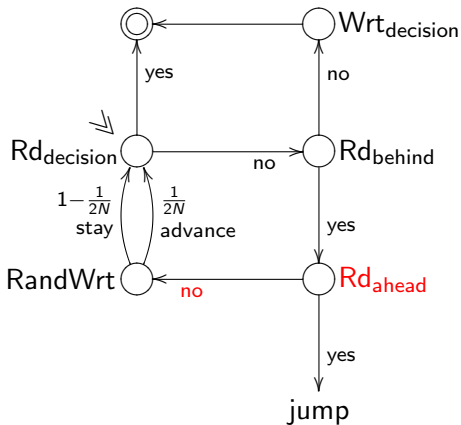
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	0
0	<u>0</u>
1	1



Example: Binary Consensus with 4 Processes

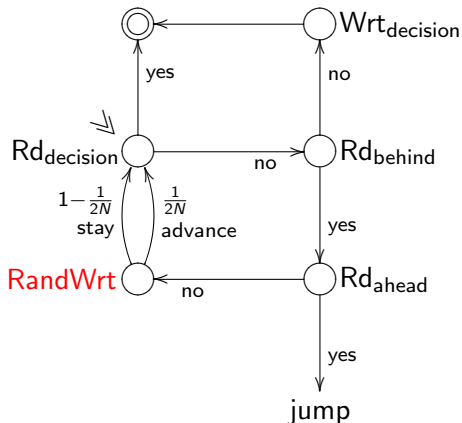
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	0
0	0
1	1



Example: Binary Consensus with 4 Processes

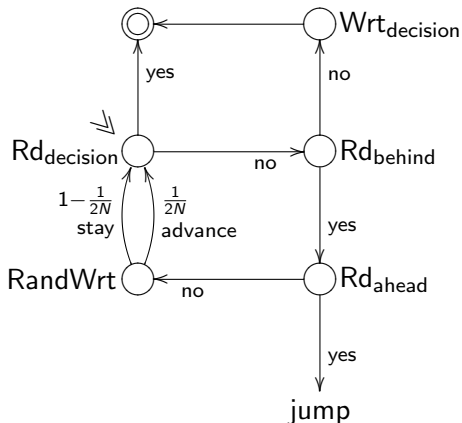
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	0
0	<u>1</u>
1	1



Example: Binary Consensus with 4 Processes

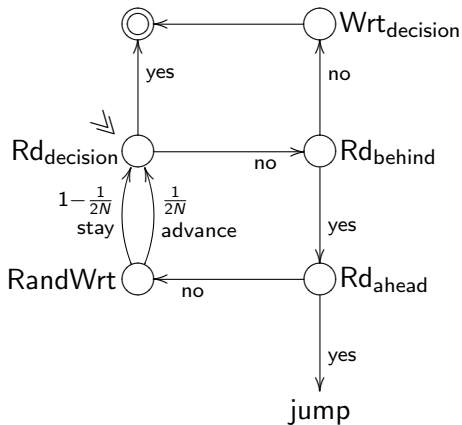
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 0, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	<u>1</u>
0	1
1	1



Example: Binary Consensus with 4 Processes

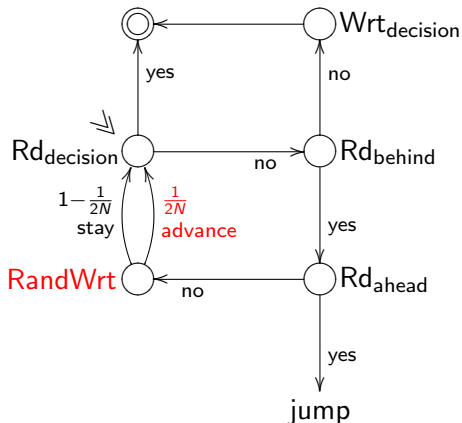
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 1, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	1
<u>1</u>	1
1	1



Example: Binary Consensus with 4 Processes

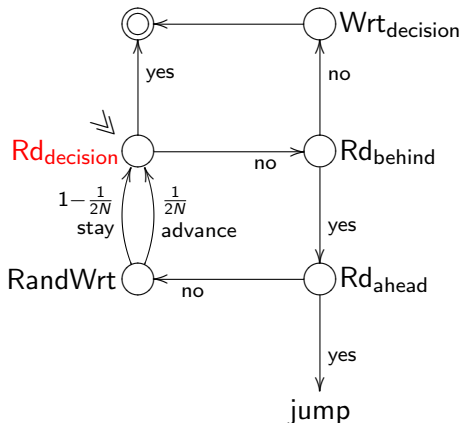
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 1, p_i = v_0, d_i = \perp$

v_0	v_1
<u>0</u>	0
0	0
0	0
0	1
1	1
1	1



Example: Binary Consensus with 4 Processes

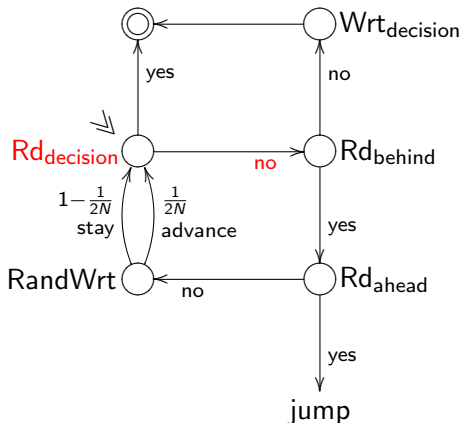
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 1, p_i = v_0, d_i = \perp$

v_0	v_1
0	<u>0</u>
0	0
0	0
0	1
1	1
1	1



Example: Binary Consensus with 4 Processes

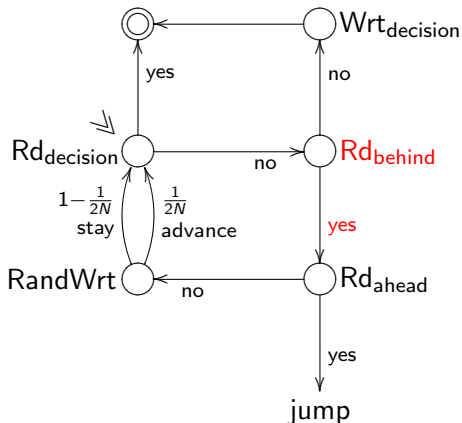
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 1, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	1
1	1
1	<u>1</u>



Example: Binary Consensus with 4 Processes

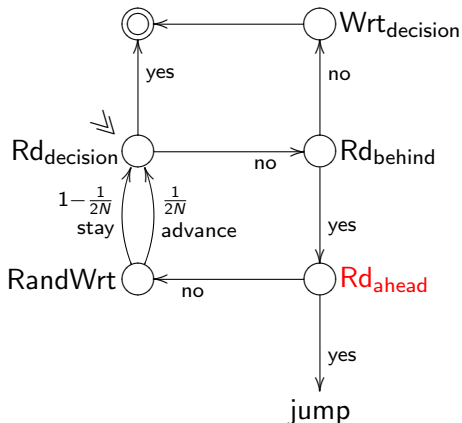
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 1, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
<u>0</u>	1
1	1
1	1



Example: Binary Consensus with 4 Processes

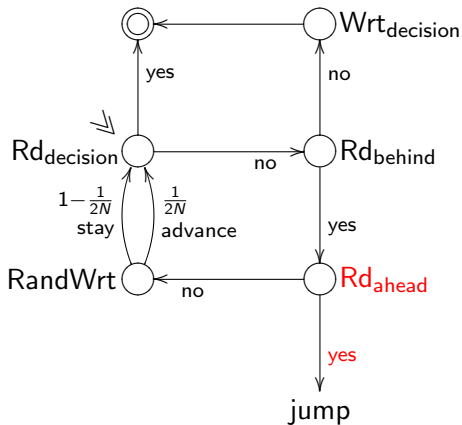
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 1, p_i = v_0, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	<u>1</u>
1	1
1	1



Example: Binary Consensus with 4 Processes

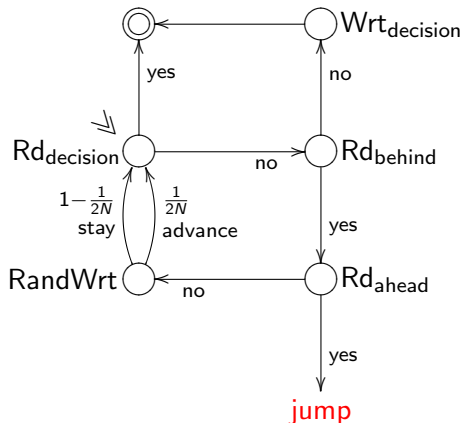
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 2, p_i = v_1, d_i = \perp$

v_0	v_1
0	0
0	0
0	0
0	1
1	1
1	1



Example: Binary Consensus with 4 Processes

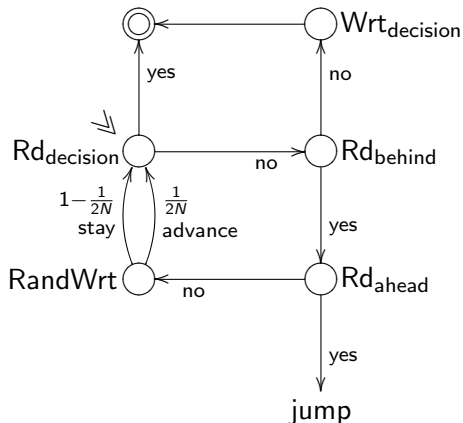
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 2, p_i = v_1, d_i = \perp$

v_0	v_1
0	0
0	0
0	<u>1</u>
0	1
1	1
1	1



Example: Binary Consensus with 4 Processes

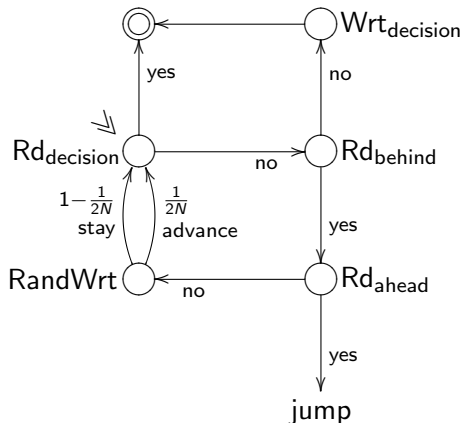
$K \times R$ registers

$K = 2$

$R = 2 \log N + 2 = 6$

$r_i = 2, p_i = v_1, d_i = \perp$

v_0	v_1
0	<u>1</u>
0	0
0	1
0	1
1	1
1	1



Correctness: Validity and Agreement

Validity

Easy: a value moves ahead only if supported by some process.

Correctness: Validity and Agreement

Validity

Easy: a value moves ahead only if supported by some process.

Agreement

$$s \models \Phi(v, v', r)$$

“In state s , the value v *eliminates* the value v' in round r .”

Correctness: Validity and Agreement

Validity

Easy: a value moves ahead only if supported by some process.

Agreement

$$s \models \Phi(v, v', r)$$

“In state s , the value v *eliminates* the value v' in round r .”

v_0	v_1
0	0
0	0
0	0
0	1
0	1
1	1

Correctness: Validity and Agreement

Validity

Easy: a value moves ahead only if supported by some process.

Agreement

$$s \models \Phi(v, v', r)$$

"In state s , the value v *eliminates* the value v' in round r ."

Proof by contradiction: disagreement implies two distinct values eliminate each other.

v_0	v_1
0	0
0	0
0	0
0	1
0	1
1	1

Correctness: Probabilistic Termination

Start from any reachable state, with highest occupied round r .

Correctness: Probabilistic Termination

Start from any reachable state, with highest occupied round r .

Consider events E_1 and E_2 :

E_1 : “a success occurs before $5N$ attempts to move from r to $r + 1$ are made and all subsequent such attempts fail;”

Correctness: Probabilistic Termination

Start from any reachable state, with highest occupied round r .

Consider events E_1 and E_2 :

E_1 : “a success occurs before $5N$ attempts to move from r to $r + 1$ are made and all subsequent such attempts fail;”

E_2 : “a success occurs before $5N$ attempts to move from $r + 1$ to $r + 2$ are made.”

Correctness: Probabilistic Termination

Start from any reachable state, with highest occupied round r .

Consider events E_1 and E_2 :

E_1 : “a success occurs before $5N$ attempts to move from r to $r + 1$ are made and all subsequent such attempts fail;”

E_2 : “a success occurs before $5N$ attempts to move from $r + 1$ to $r + 2$ are made.”

Claims:

- $E_1 \wedge E_2 \Rightarrow$ “at least one process terminates successfully in round $r + 2$ before $15N$ complete loops are executed.”

Correctness: Probabilistic Termination

Start from any reachable state, with highest occupied round r .

Consider events E_1 and E_2 :

E_1 : “a success occurs before $5N$ attempts to move from r to $r + 1$ are made and all subsequent such attempts fail;”

E_2 : “a success occurs before $5N$ attempts to move from $r + 1$ to $r + 2$ are made.”

Claims:

- $E_1 \wedge E_2 \Rightarrow$ “at least one process terminates successfully in round $r + 2$ before $15N$ complete loops are executed.”
- $\mathbf{P}[E_1 \wedge E_2] \geq 0.511$.

Correctness: Probabilistic Termination

Start from any reachable state, with highest occupied round r .

Consider events E_1 and E_2 :

E_1 : “a success occurs before $5N$ attempts to move from r to $r + 1$ are made and all subsequent such attempts fail;”

E_2 : “a success occurs before $5N$ attempts to move from $r + 1$ to $r + 2$ are made.”

Claims:

- $E_1 \wedge E_2 \Rightarrow$ “at least one process terminates successfully in round $r + 2$ before $15N$ complete loops are executed.”
- $\mathbf{P}[E_1 \wedge E_2] \geq 0.511$.
- Wait-free; $O(N \log(\log N))$.

Probabilistic Symbolic Model Checker (PRISM)

Input language: based on *reactive modules* of [Alur and Henzinger, 1999].

- *Modules*: variable declarations and commands.

Probabilistic Symbolic Model Checker (PRISM)

Input language: based on *reactive modules* of [Alur and Henzinger, 1999].

- *Modules*: variable declarations and commands.
- Each command has a *guard* and a finite number of *updates*.

Probabilistic Symbolic Model Checker (PRISM)

Input language: based on *reactive modules* of [Alur and Henzinger, 1999].

- *Modules*: variable declarations and commands.
- Each command has a *guard* and a finite number of *updates*.
- Communication via global variables or action synchronization.

Probabilistic Symbolic Model Checker (PRISM)

Input language: based on *reactive modules* of [Alur and Henzinger, 1999].

- *Modules*: variable declarations and commands.
- Each command has a *guard* and a finite number of *updates*.
- Communication via global variables or action synchronization.

In our model:

- shared memory modeled as global variables (so **no** action synchronization);

Probabilistic Symbolic Model Checker (PRISM)

Input language: based on *reactive modules* of [Alur and Henzinger, 1999].

- *Modules*: variable declarations and commands.
- Each command has a *guard* and a finite number of *updates*.
- Communication via global variables or action synchronization.

In our model:

- shared memory modeled as global variables (so **no** action synchronization);
- trivial to encode atomic random-write assumption.

Probabilistic Symbolic Model Checker (PRISM)

Underlying model: Markov Decision Processes (MDP).

Specification language: Probabilistic Computation Tree Logic (PCTL).

Probabilistic Symbolic Model Checker (PRISM)

Underlying model: Markov Decision Processes (MDP).

Specification language: Probabilistic Computation Tree Logic (PCTL).

Example:

$P_{\min}=? [\text{true} \cup ((s_0=7) \mid (s_1=7) \mid (s_2=7) \mid (s_3=7))]$

PRISM returns the minimum probability that “eventually at least one process decides.”

Model Checking Results

N	R	#Phases	Model Construction		Agreement
			#States	Time(sec)	Time(sec)
2	2	30	42,320	4	0.025
3	4	90	12,280,910	213	0.094
4	2	60	45,321,126	429	0.078
4	4	40	377,616,715	5224	3.926

Model Checking Results

N	R	#Phases	Model Construction		Agreement
			#States	Time(sec)	Time(sec)
2	2	30	42,320	4	0.025
3	4	90	12,280,910	213	0.094
4	2	60	45,321,126	429	0.078
4	4	40	377,616,715	5224	3.926

N	R	#Phases	Probabilistic Termination		
			Time(sec)	MinProb	AnalyticBd
2	2	30	6	0.745	0.511
3	4	90	2,662	0.971	0.667
4	2	60	602	0.755	0.511
4	4	40	55,795	0.765	0.750

Concluding Remarks

MWMM Memory

Concluding Remarks

MWMM Memory

- “comparable” to SWMM: both involve $O(N)$ slowdown if implemented from SWSR.

Concluding Remarks

MWMM Memory

- “comparable” to SWMR: both involve $O(N)$ slowdown if implemented from SWSR.
- Space requirement: $O(\log N)$ registers of one bit each.

Concluding Remarks

MWMM Memory

- “comparable” to SWMM: both involve $O(N)$ slowdown if implemented from SWSR.
- Space requirement: $O(\log N)$ registers of one bit each.
- Processes anonymous.

Concluding Remarks

MWMM Memory

- “comparable” to SWMM: both involve $O(N)$ slowdown if implemented from SWSR.
- Space requirement: $O(\log N)$ registers of one bit each.
- Processes anonymous.
- Reduced data: each memory access carries one bit.

Concluding Remarks

MWMMR Memory

- “comparable” to SWMR: both involve $O(N)$ slowdown if implemented from SWSR.
- Space requirement: $O(\log N)$ registers of one bit each.
- Processes anonymous.
- Reduced data: each memory access carries one bit.
- Model checking feasible.

Concluding Remarks

Per Process Work

- Expected work for isolated process: $\Omega(N)$.

Concluding Remarks

Per Process Work

- Expected work for isolated process: $\Omega(N)$.
- Can this be reduced to $O(\log N)$?

Concluding Remarks

Per Process Work

- Expected work for isolated process: $\Omega(N)$.
- Can this be reduced to $O(\log N)$?

Comments on PRISM

- Minimal learning effort, so useful for rapid prototyping.

Concluding Remarks

Per Process Work

- Expected work for isolated process: $\Omega(N)$.
- Can this be reduced to $O(\log N)$?

Comments on PRISM

- Minimal learning effort, so useful for rapid prototyping.
- Awaiting symmetry reduction ...

Concluding Remarks

Per Process Work

- Expected work for isolated process: $\Omega(N)$.
- Can this be reduced to $O(\log N)$?

Comments on PRISM

- Minimal learning effort, so useful for rapid prototyping.
- Awaiting symmetry reduction ...

– End –

Other Weak-Adversary Algorithms

Write-oblivious: unread register content hidden from adversary.

- Chandra, 1996: $O(N \log^2 N)$, MWMR
- Aumann, 1997: $O(N \log^4 N)$, SWMR; $O(N \log N)$, MWMR

Value-oblivious: all parameter values hidden from adversary.

- Aumann and Kapah-Levy, 1999: $O(N \log N \cdot e^{\sqrt{\log N}})$, SWSR
- Aumann and Bender, 2004: $O(N \log^2 N)$, MWMR

Oblivious: predetermined list of process names, independent of dynamic random choices.

- Aumann, Bender and Zhang, 1997: $O(N \log N \log(\log N))$ for N processes *and* N words, MWMR