Molecular Computation
An Algorithmic Approach

Rati Gelashvili

Joint work with
Dan Alistarh (ETH), David Eisenstat (Google),
James Aspnes (Yale), Milan Vojnovic (MSR), Ron Rivest (MIT)

Distributed Systems

Ingredients:

Distributed Systems

Ingredients:

- Nodes

Distributed Systems

Ingredients:

- Nodes
- Communication

Distributed Systems

Ingredients:

- Nodes
- Communication
- Computation

Computational Model

Population Protocols [AADFP'04]

Computational Model
Population Protocols [AADFP'04]

- **Nodes** are ***simple, identical agents***
  - Each node is ***the same*** finite state automaton
  - For example: a molecule

Computational Model
Population Protocols [AADFP'04]

- **Nodes** are ***simple***, ***identical agents***
  - Each node is ***the same*** finite state automaton
  - For example: a molecule

- **Interactions** are ***pairwise***, and follow a ***fair scheduler***
  - Usually considered ***uniform random***
  - Nodes update their state following interactions

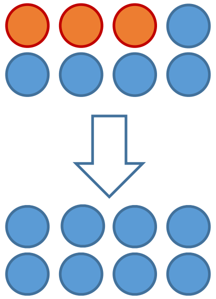
Computational Model

Population Protocols [AADFP'04]

- **Nodes** are ***simple***, ***identical agents***
  - Each node is ***the same*** finite state automaton
  - For example: a molecule

- **Interactions** are ***pairwise***, and follow a ***fair scheduler***
  - Usually considered ***uniform random***
  - Nodes update their state following interactions

- **Computation** is performed ***collectively***
  - The system ***should converge*** to configurations satisfying meaningful predicates
  - No "fixed" decision time

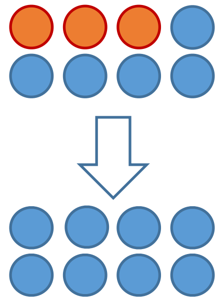


Computational Model
Population Protocols [AADFP’04]

- **Nodes** are ***simple***, ***identical agents***
  - Each node is ***the same*** finite state automaton
  - For example: a molecule

- **Interactions** are ***pairwise***, and follow a ***fair scheduler***
  - Usually considered ***uniform random***
  - Nodes update their state following interactions

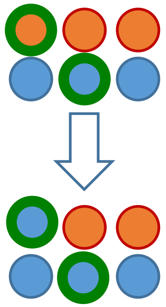
- **Computation** is performed ***collectively***
  - The system ***should converge*** to configurations satisfying meaningful predicates
  - No “fixed” decision time
- A.k.a. Chemical Reaction Networks



Complexity

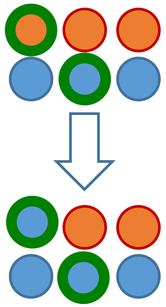
1. Time

- **Round** = a ***single pair*** interacts
  - Chosen uniformly at random
- **Parallel convergence time**
  - **#rounds** to convergence / **# nodes**
  - Alternative continuous-time definition exists



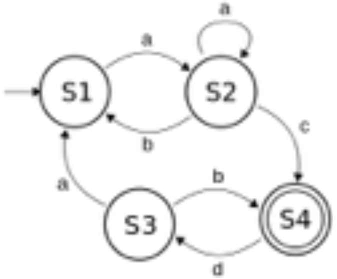
Complexity

1. Time



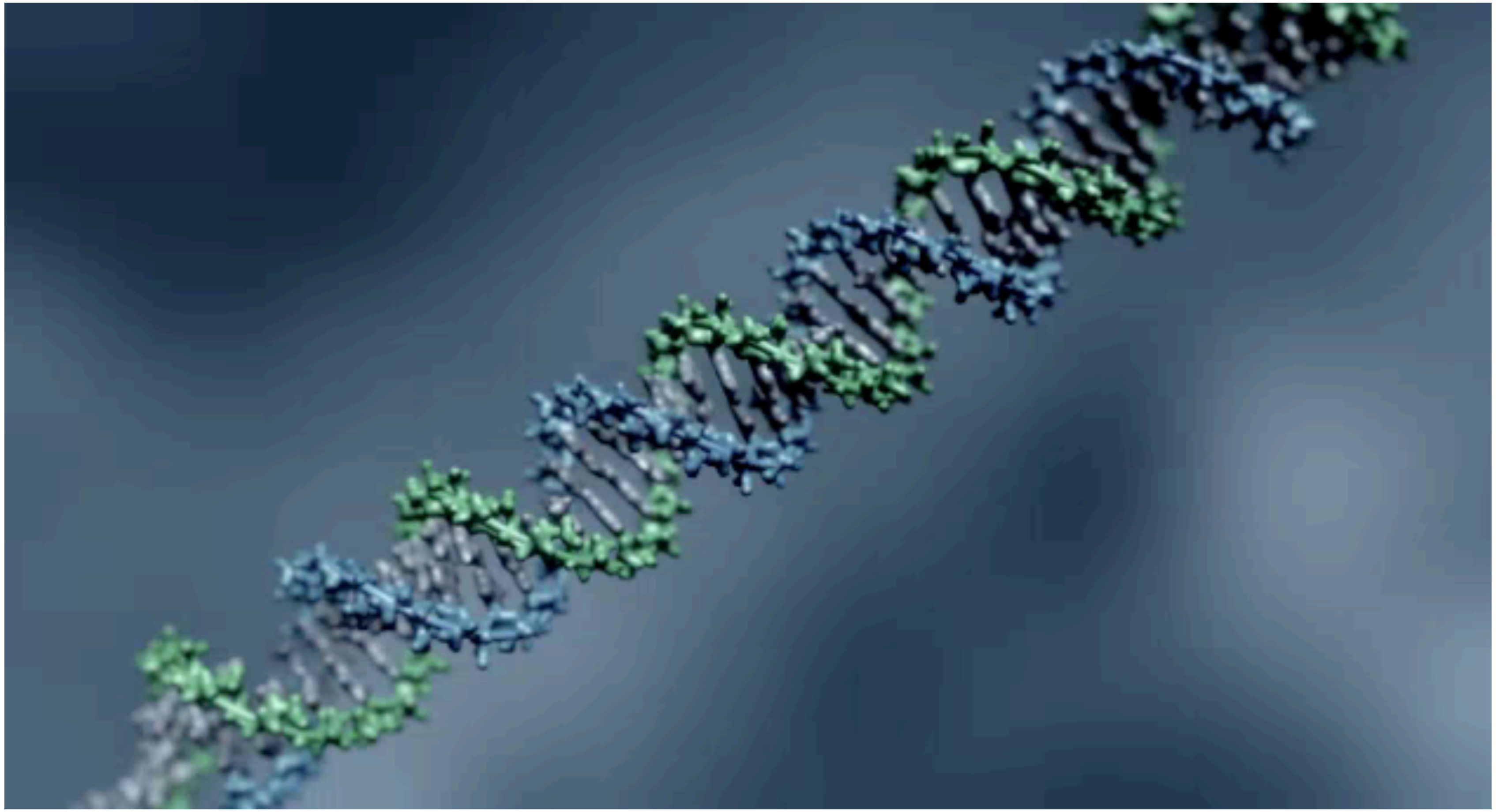
- **Round** = a ***single pair*** interacts
  - Chosen uniformly at random
- **Parallel convergence time**
  - **#rounds** to convergence / **# nodes**
  - Alternative continuous-time definition exists

2. Space



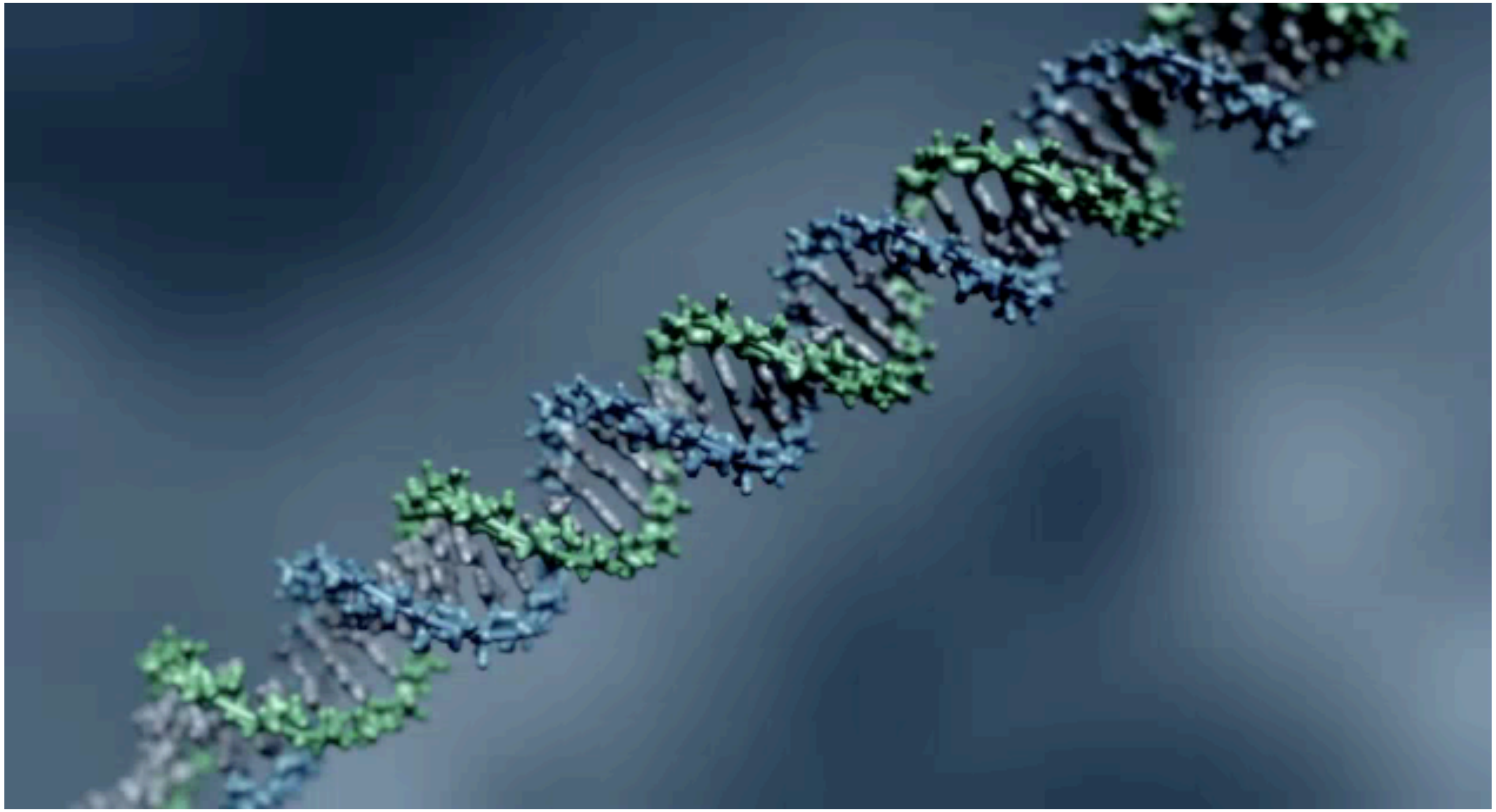
- Number of ***distinct states*** per automaton
- Alternatively, #**memory bits** to encode state

More Precisely: Communication



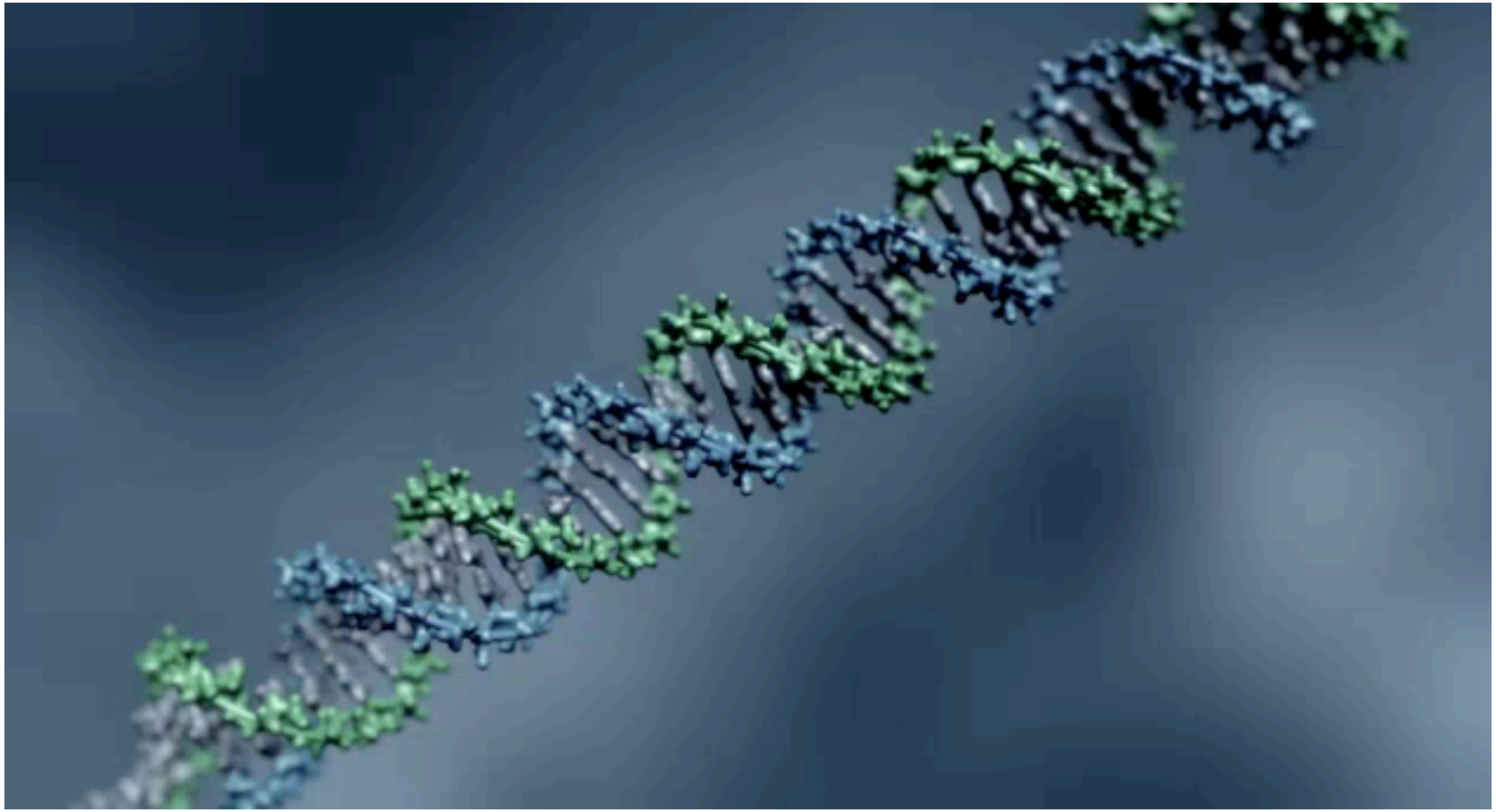
Courtesy of the Microsoft Research Biological Computation Group

More Precisely: Communication



Courtesy of the Microsoft Research Biological Computation Group

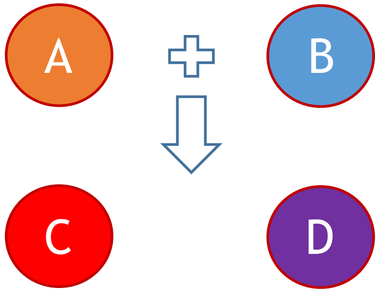
More Precisely: Communication



Courtesy of the Microsoft Research Biological Computation Group

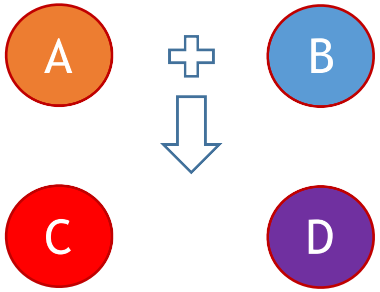
What can we compute?

We can perform interactions of the type:



What can we compute?

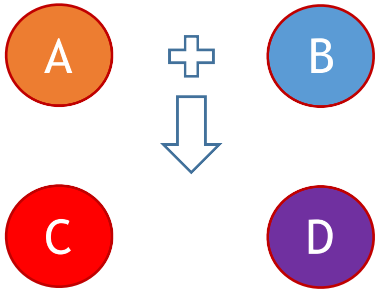
We can perform interactions of the type:



Example: the **OR** function

- **Initial** states: 0 or 1
- **Final** state:
  - If there exists a 1, then all 1.
  - Otherwise, all 0
- Protocol:

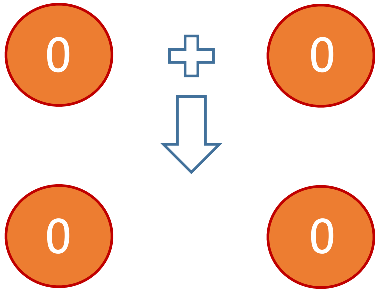
What can we compute?



We can perform interactions of the type:

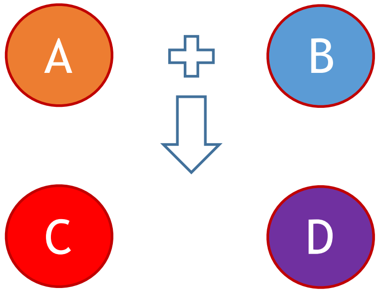
Example: the **OR** function

- **Initial** states: 0 or 1
- **Final** state:
  - If there exists a 1, then all 1.
  - Otherwise, all 0
- Protocol:



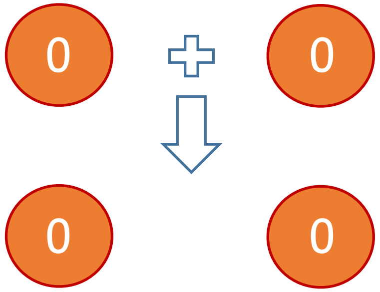
What can we compute?

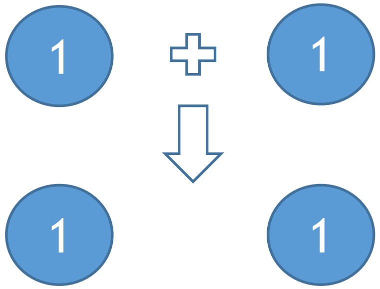
We can perform interactions of the type:



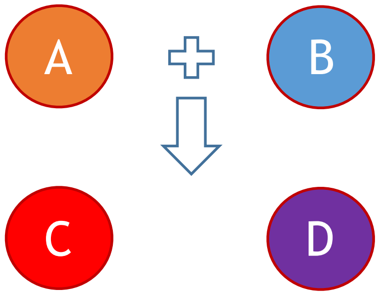
Example: the **OR** function

- **Initial** states: 0 or 1
- **Final** state:
    - If there exists a 1, then all 1.
    - Otherwise, all 0
- Protocol:





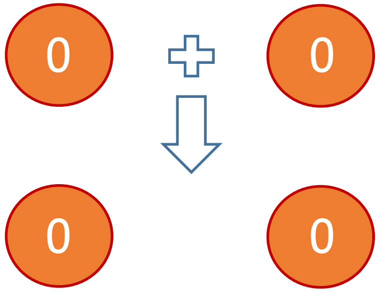
What can we compute?

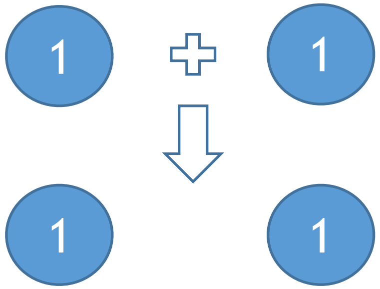


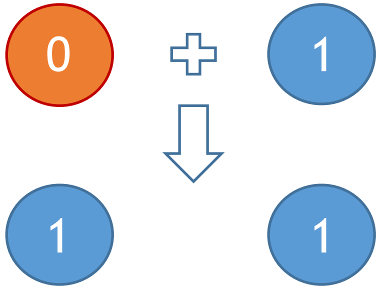
We can perform interactions of the type:

Example: the **OR** function

- **Initial** states: 0 or 1
- **Final** state:
  - If there exists a 1, then all 1.
  - Otherwise, all 0
- Protocol:

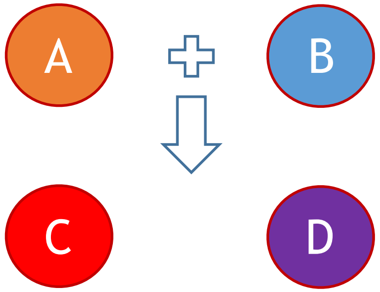






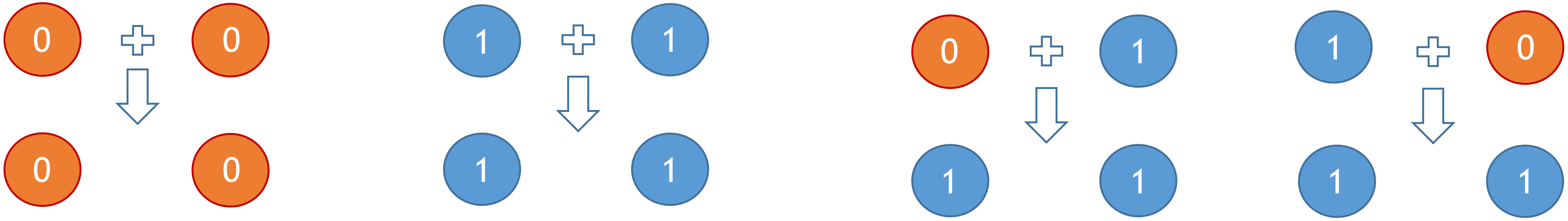
What can we compute?

We can perform interactions of the type:



Example: the **OR** function

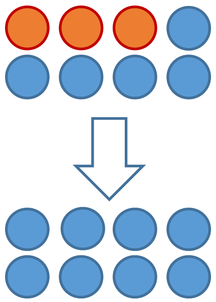
- **Initial** states: 0 or 1
- **Final** state:
  - If there exists a 1, then all 1.
  - Otherwise, all 0
- Protocol:



The Majority Function

Majority ("Consensus")

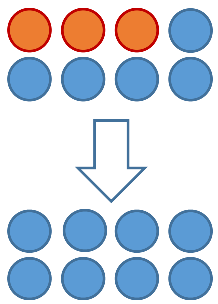
- **Initial** states A, B
- **Output**:
  - **A** if **#A** > **#B** initially.
  - **B**, otherwise.



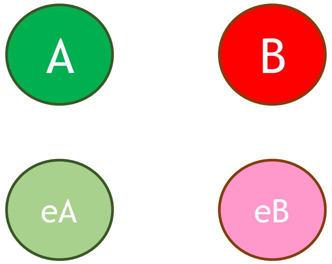
The Majority Function

Majority ("Consensus")

- **Initial** states A, B
- **Output**:
  - **A** if **#A** > **#B** initially.
  - **B**, otherwise.
- **Fundamental task**
  - **Complexity**: [AAE08] & [DV12]; [PVV09] & [MNRS14]
  - Natural computation:
    the cell cycle switch implements approximate majority [CC12]
  - Implementation in DNA: [CDS+13, Nature Nanotechnology]



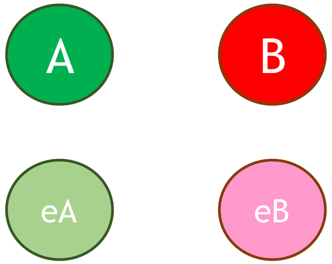
Solving Majority



4-State Exact Majority [PVV09] [MNRS14]

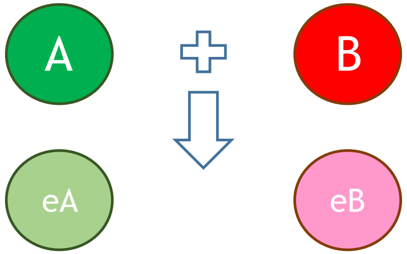
- Protocol:

Solving Majority

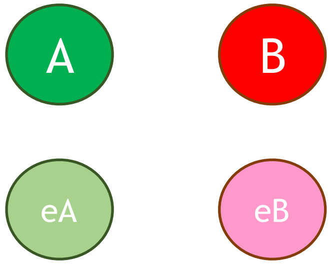


4-State Exact Majority [PVV09] [MNRS14]

- Protocol:

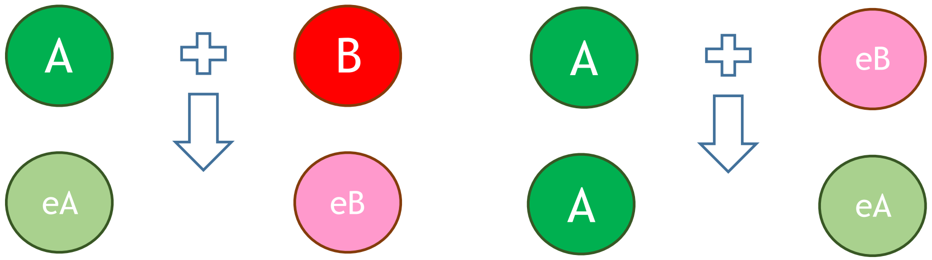


Solving Majority

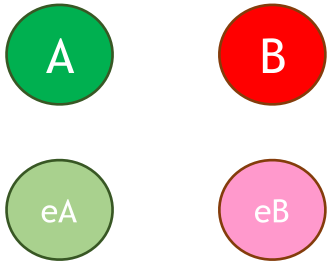


4-State Exact Majority [PVV09] [MNRS14]

- Protocol:

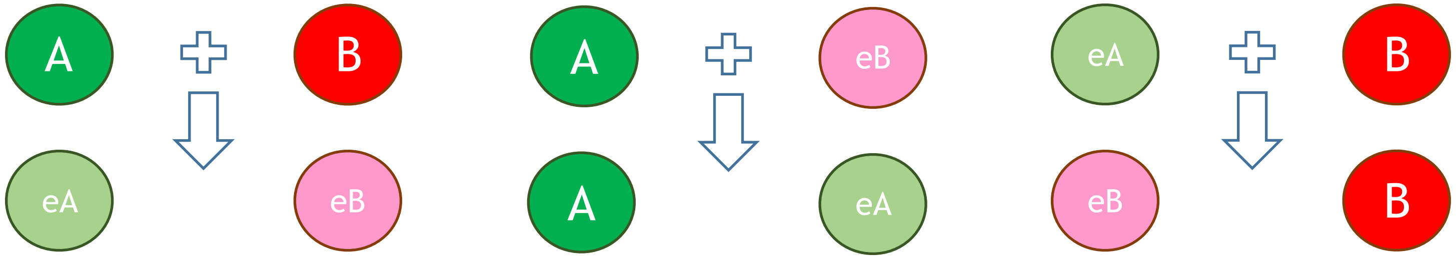


Solving Majority

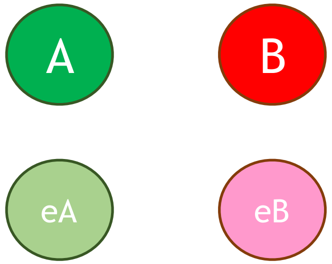


4-State Exact Majority [PVV09] [MNRS14]

- Protocol:

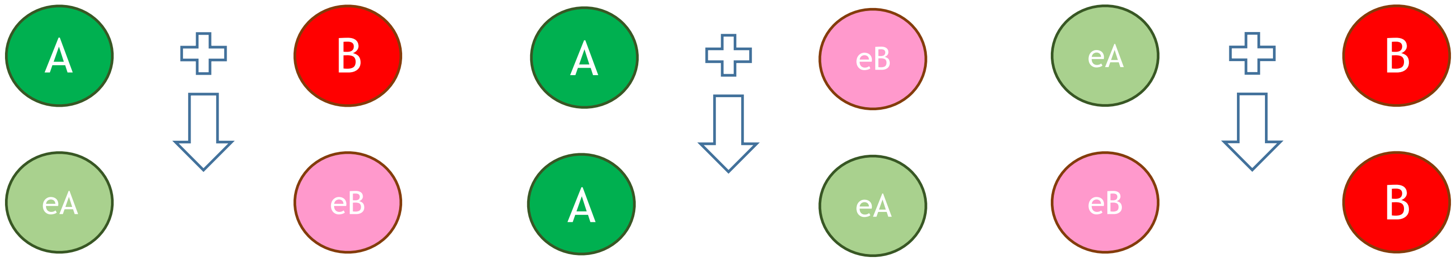


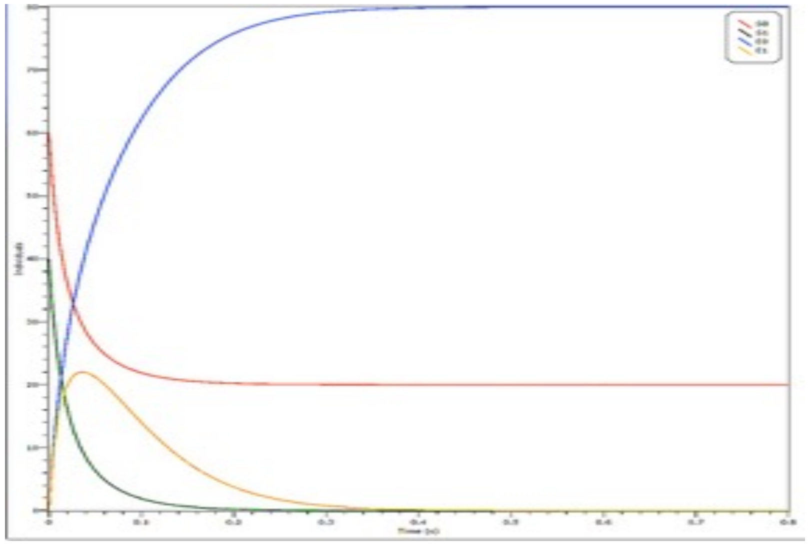
Solving Majority



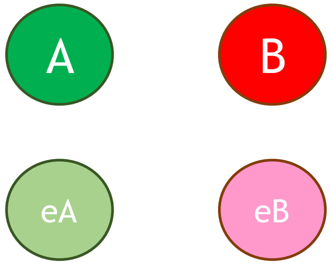
4-State Exact Majority [PVV09] [MNRS14]

- Protocol:



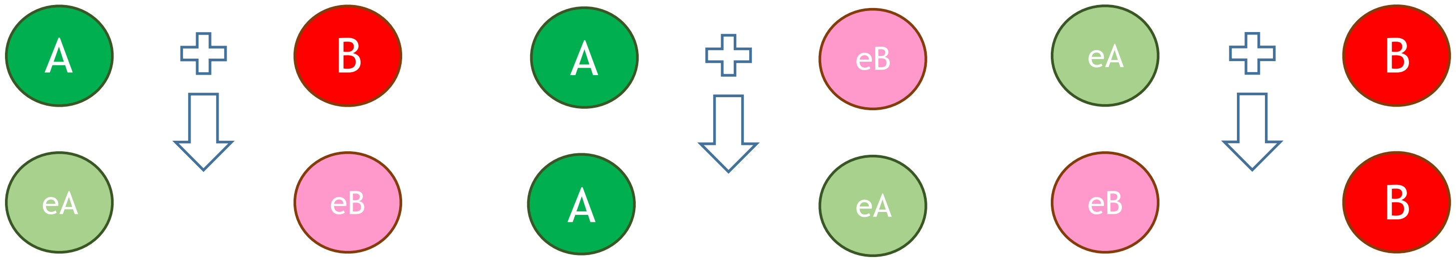


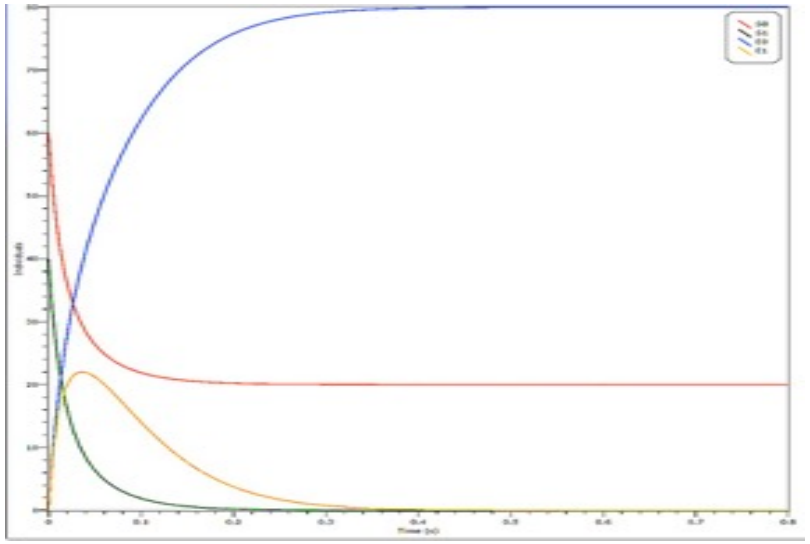
Solving Majority



4-State Exact Majority [PVV09] [MNRS14]

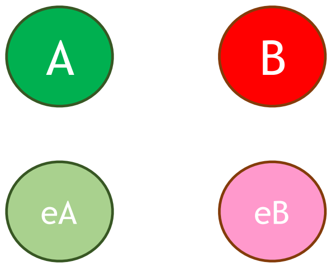
- Protocol:





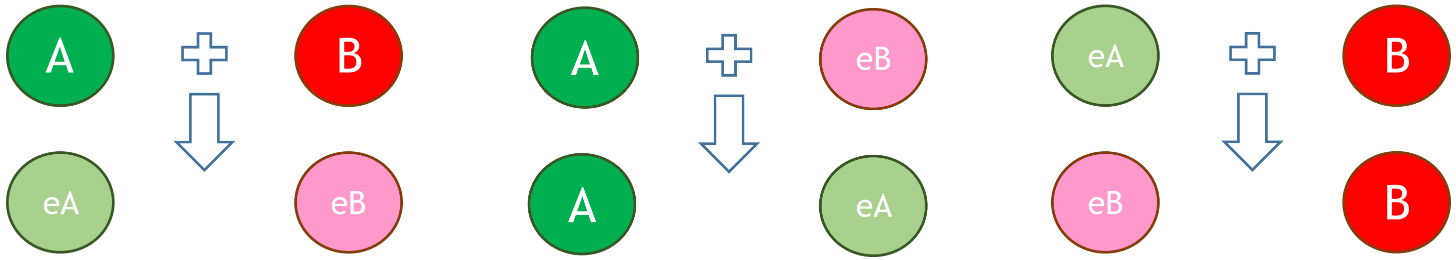
Discrepancy/margin:
$\varepsilon = |\#A - \#B| / n$
Can be as small as
$\varepsilon = O(1 / n)$.

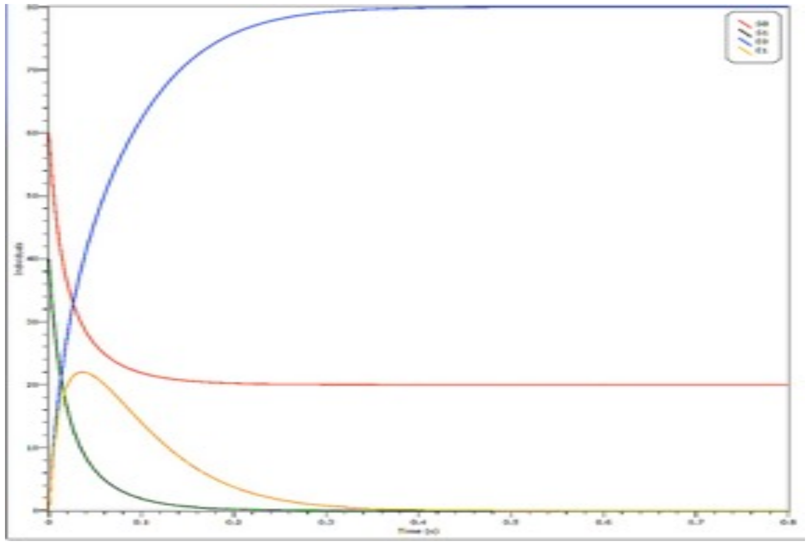
Solving Majority



4-State Exact Majority [PVV09] [MNRS14]

- Protocol:

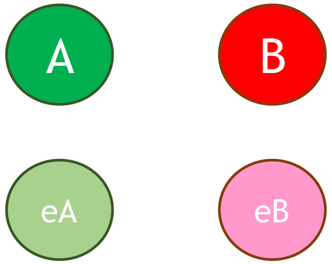




Discrepancy/margin:
$\varepsilon = |\#A - \#B| / n$
Can be as small as
$\varepsilon = O(1 / n)$.

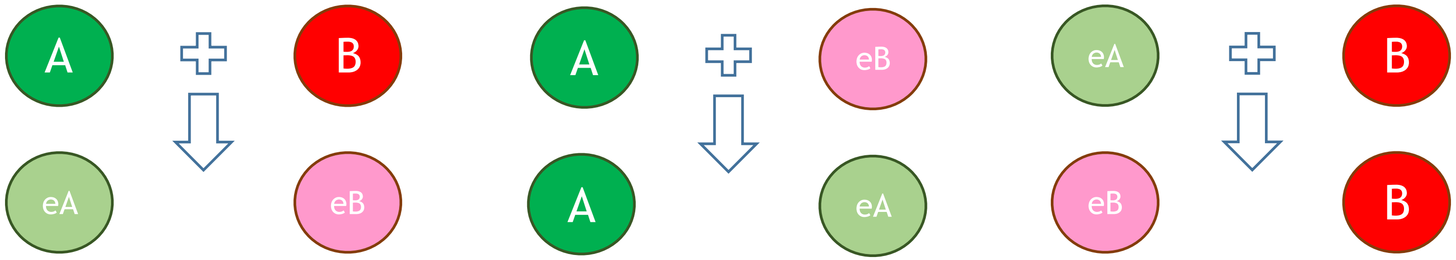
Theorem: Given n nodes and discrepancy ε, the running time of 4EM is $O((\log n) / \varepsilon)$.

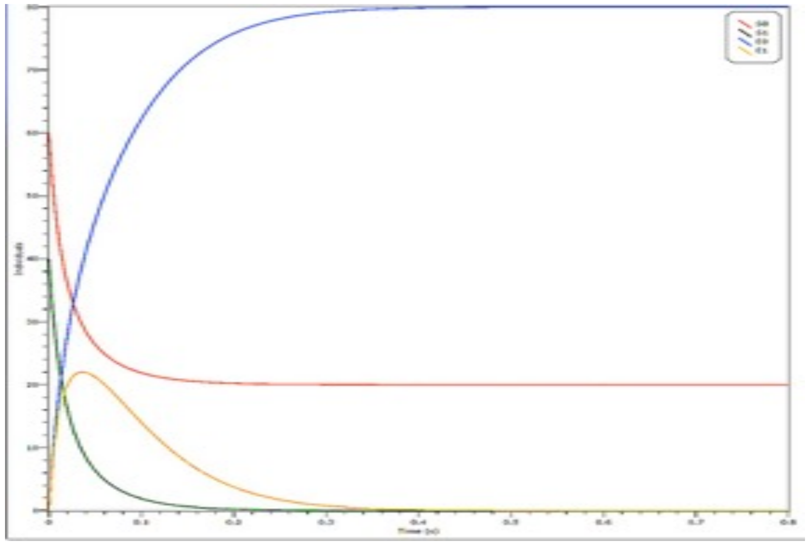
Solving Majority



4-State Exact Majority [PVV09] [MNRS14]

- Protocol:



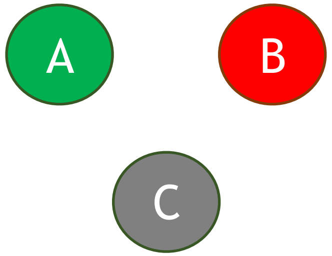


Discrepancy/margin:
$\varepsilon = |\#A - \#B| / n$
Can be as small as
$\varepsilon = O(1 / n)$.

Theorem: Given n nodes and discrepancy ε, the running time of 4EM is $O((\log n) / \varepsilon)$.

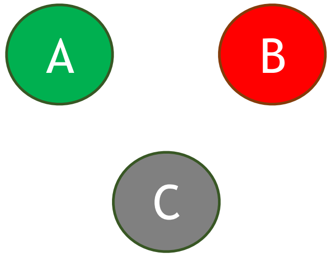
Can be $\Theta(n \log n)$ if ε = constant / n.

Solving Majority *Approximately*



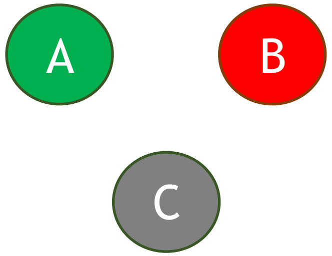
- 3-state Approximate Majority [AAE08] [DV12]

Solving Majority *Approximately*

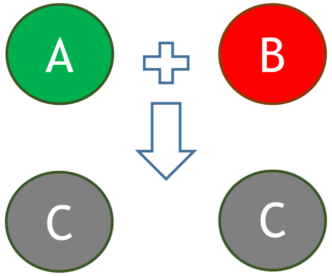


- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:

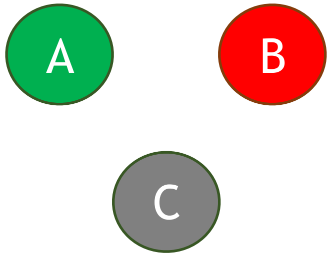
Solving Majority *Approximately*



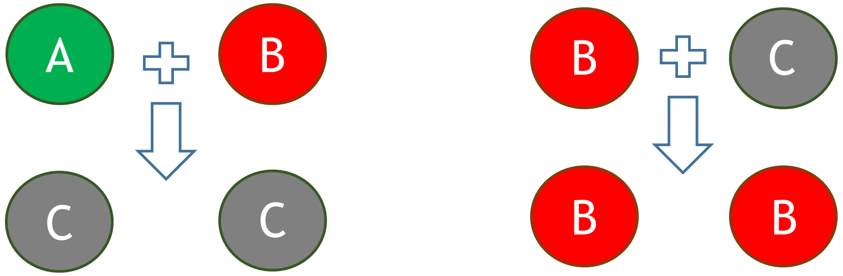
- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:



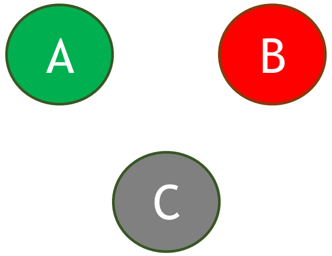
Solving Majority *Approximately*



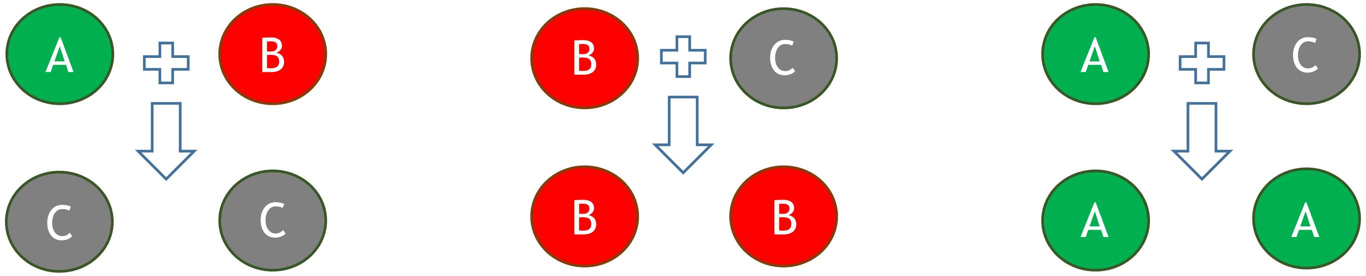
- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:



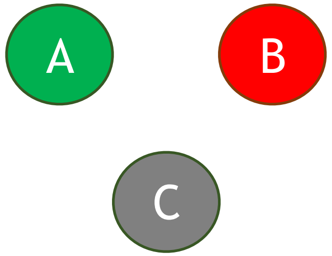
Solving Majority *Approximately*



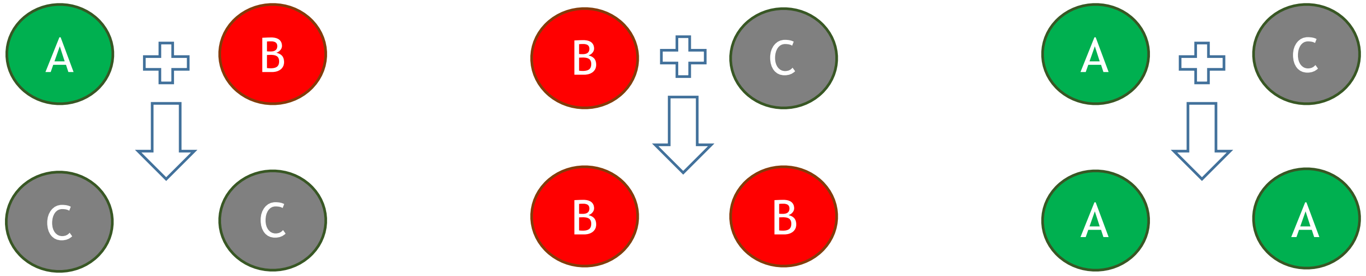
- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:



Solving Majority *Approximately*

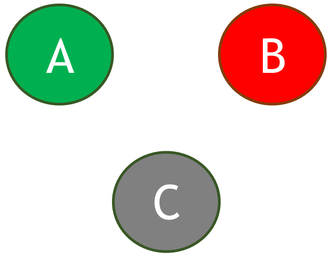


- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:

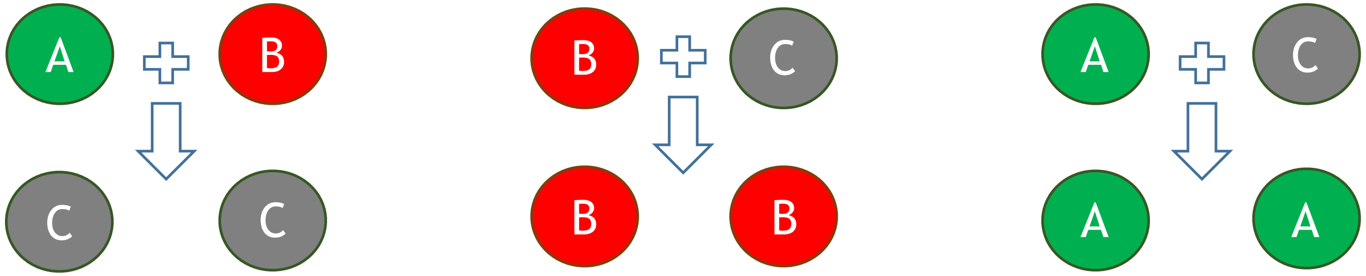


- Execution:

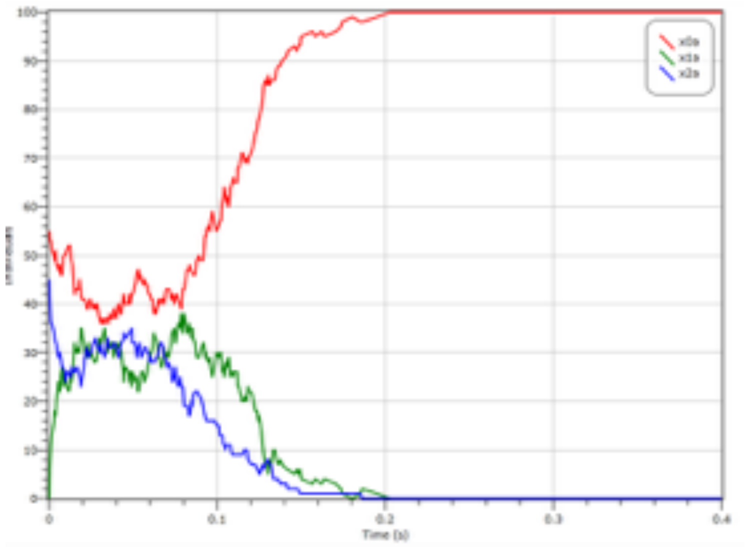
Solving Majority *Approximately*



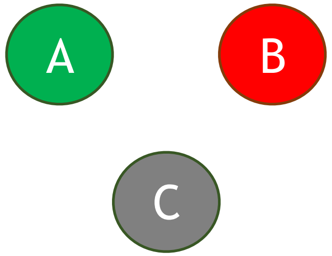
- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:



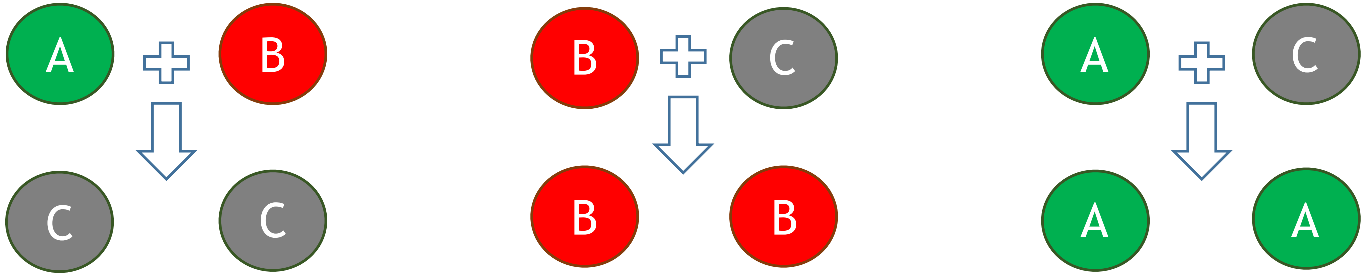
- Execution:



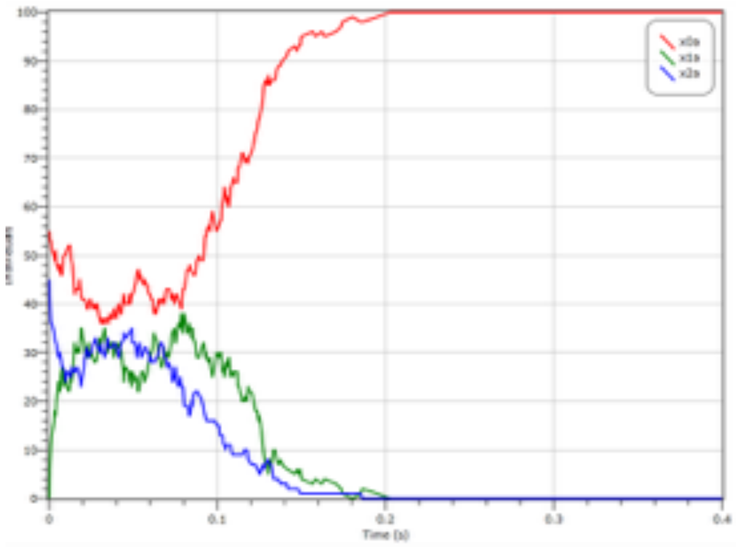
Solving Majority *Approximately*

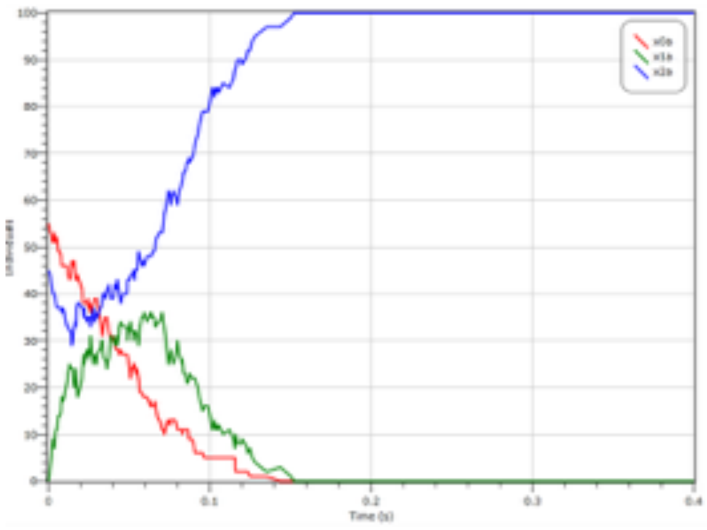


- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:

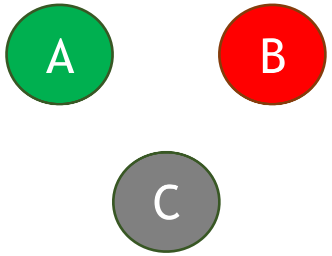


- Execution:

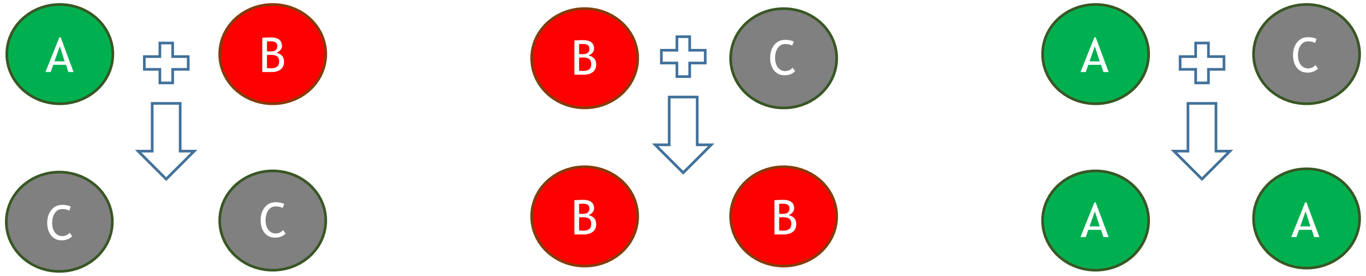




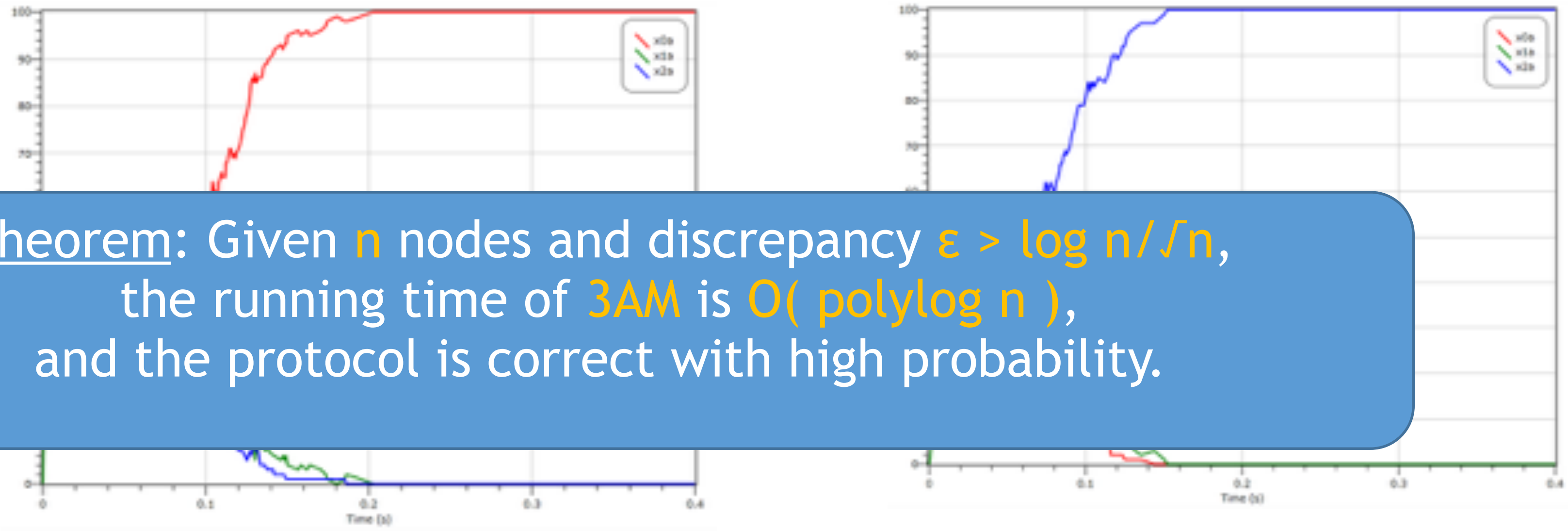
Solving Majority *Approximately*



- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:

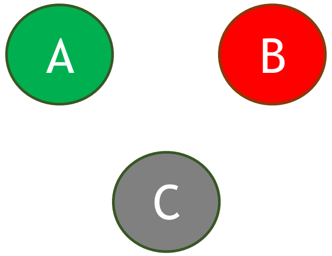


- Execution:

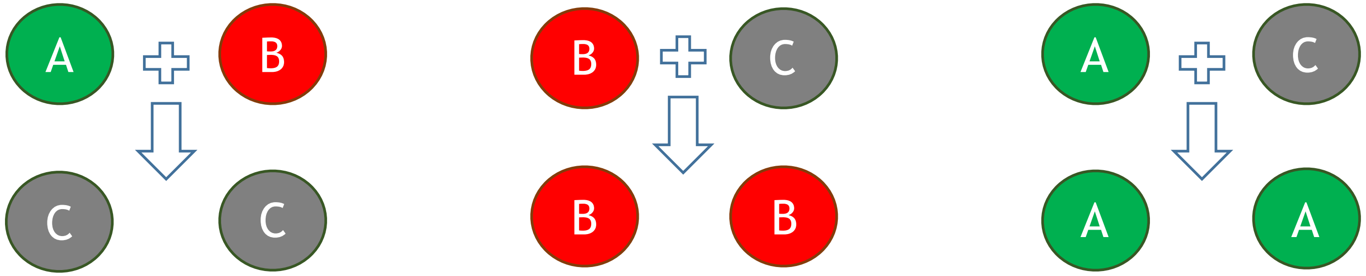


<u>Theorem</u>: Given n nodes and discrepancy $\varepsilon > \log n / \sqrt{n}$,
the running time of 3AM is $O(\text{ polylog } n \text{ })$,
and the protocol is correct with high probability.

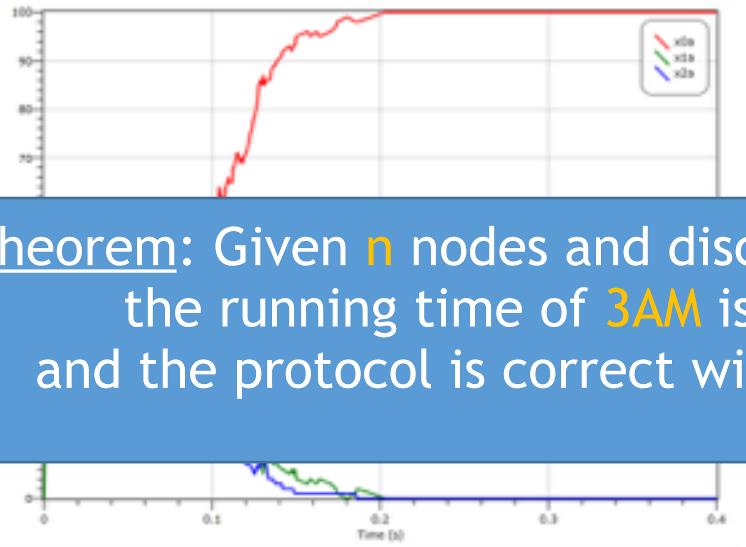
Solving Majority *Approximately*

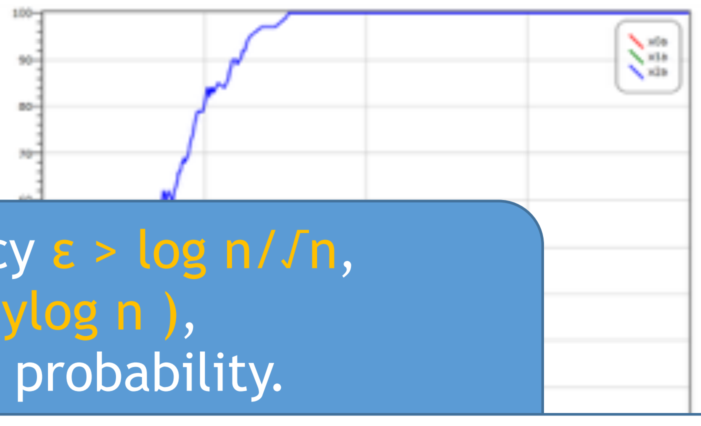


- 3-state Approximate Majority [AAE08] [DV12]
- The protocol:



- Execution:





<u>Theorem</u>: Given n nodes and discrepancy $\varepsilon > \log n/\sqrt{n}$,
the running time of 3AM is $O(\text{polylog } n)$,
and the protocol is correct with high probability.

Error probability can be as high as ***constant*** for lower discrepancy.

The Status

Algorithm	Reliability	Speed
The Four-State Protocol	Exact	Slow (super-linear)
The Three-State Protocol	Flaky (Up to Constant Error)	Fast (poly-logarithmic)

Average&Conquer

Algorithm	Reliability	Speed
The Four-State Protocol	Exact	Slow (super-linear)
The Three-State Protocol	Flaky (Up to Constant Error)	Fast (poly-logarithmic)
Average&Conquer [PODC 2015]	Exact	Fast (poly-logarithmic)

Average&Conquer

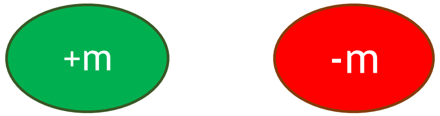
Algorithm	Reliability	Speed
The Four-State Protocol	Exact	Slow (super-linear)
The Three-State Protocol	Flaky (Up to Constant Error)	Fast (poly-logarithmic)
Average&Conquer [PODC 2015] (Super-Constant State Space)	Exact	Fast (poly-logarithmic)

The Plan

- Population Protocols
- The Majority Problem
  - 4EM
  - 3AM
  - Average-and-Conquer (AVC)
  - Quantized AVC
- Impossibility Results
- Open Questions
- Leader Election Problem

Simplified AVC: Main Ideas

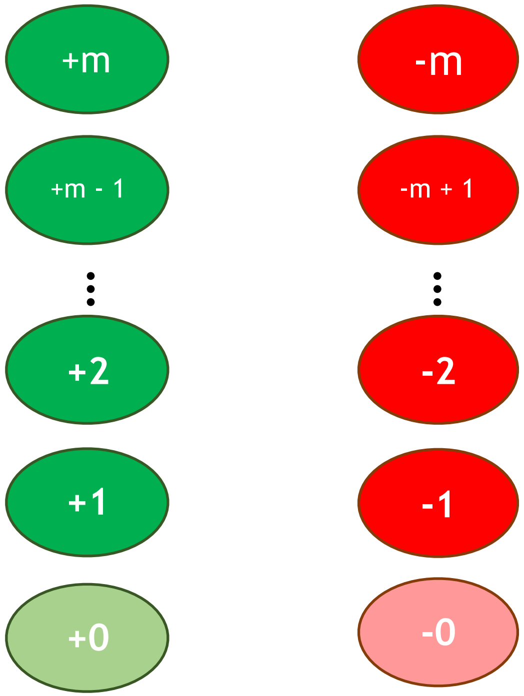
- Each **state** corresponds to a **value ("confidence level")**
  - **Strong states (non-negative value):**
    - **Positive -> A**
    - **Negative -> B**
  - **Weak: value +/- 0**
- All nodes start with absolute value **m > 0**
  - **+m** if **A**
  - **-m** if **B**



- Two interaction types:
  - **Averaging**: strong (non-zero) nodes average out their values
  - **Conquer**: strong (non-zero) nodes bring weak nodes to "their side"
- Output:
  - If **positive** or **+0**, then **A**
  - If **negative** or **-0**, then **B**

AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

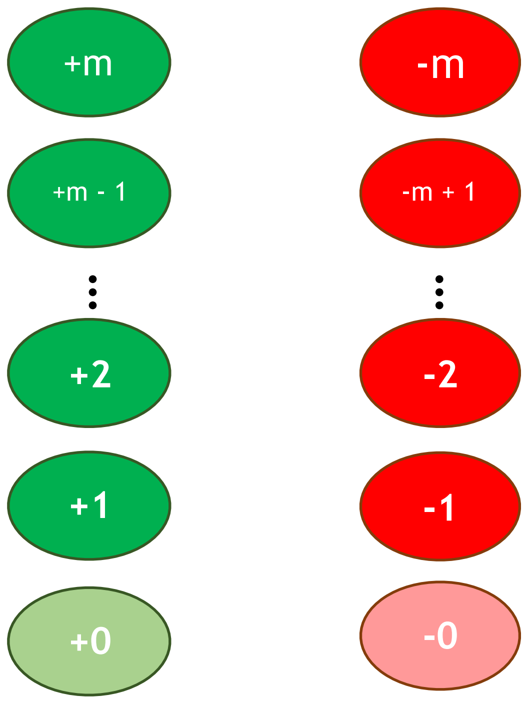


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

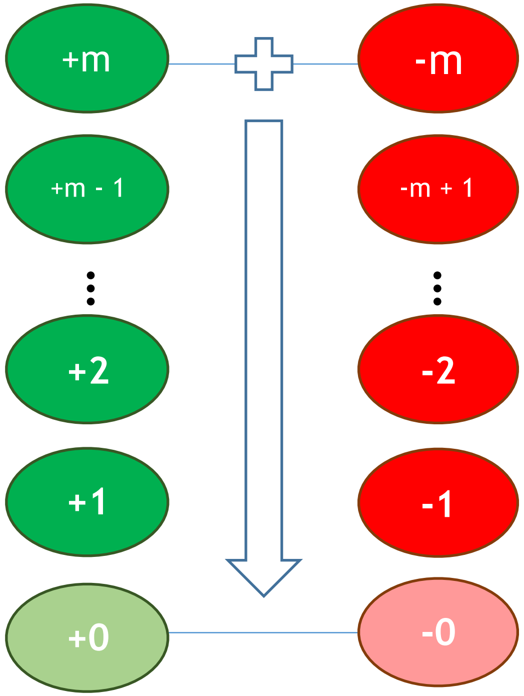


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

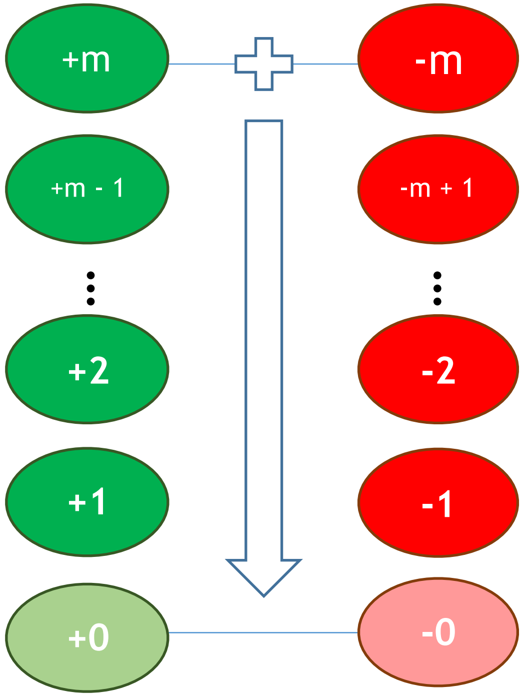


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

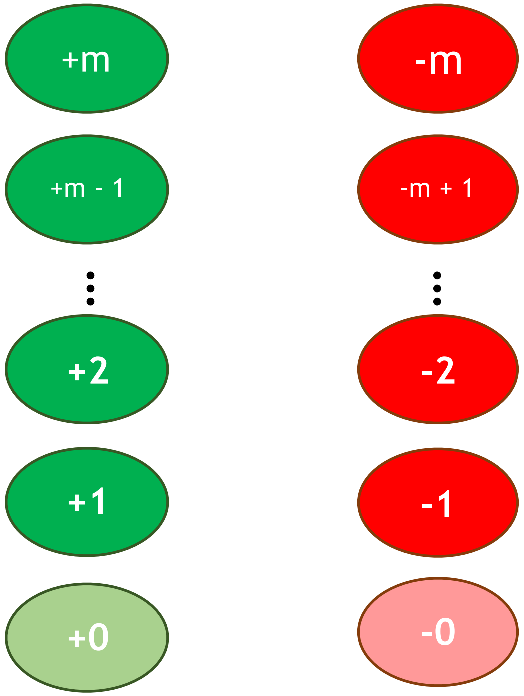


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

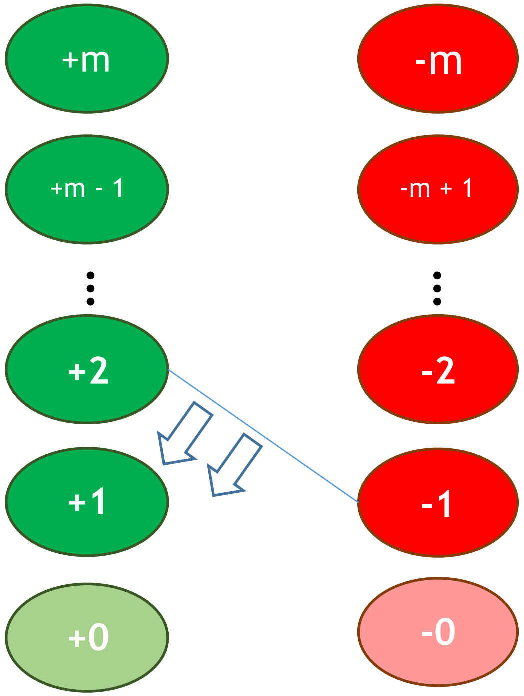


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

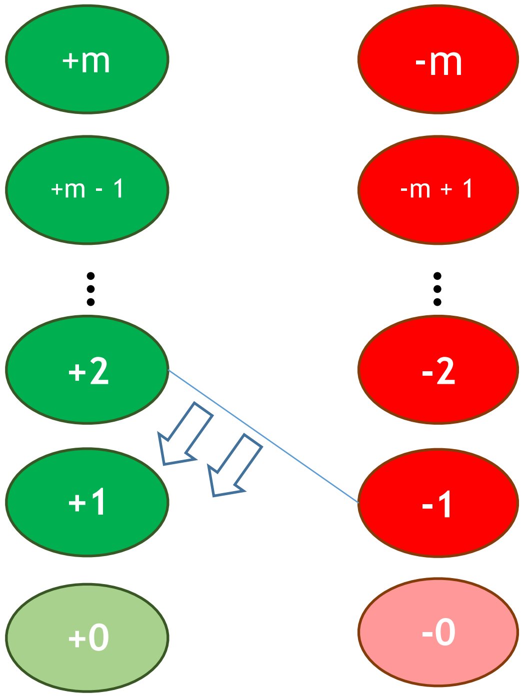
- Whenever two strong nodes meet, they average values



AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

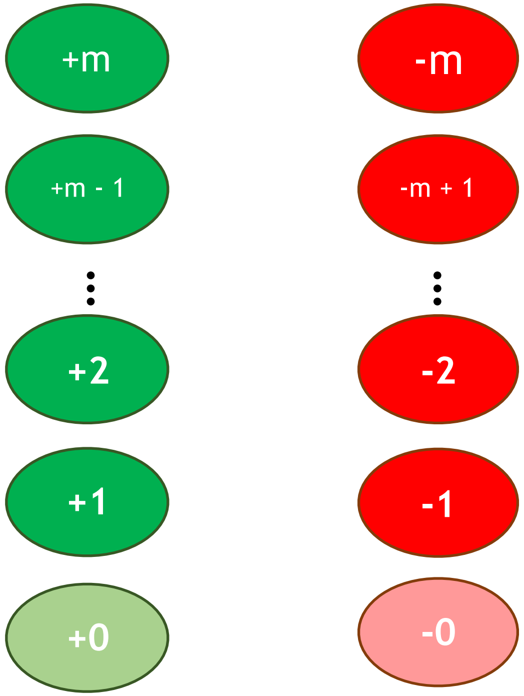
Averaging:
- Whenever two strong nodes meet, they average values



AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:
- Whenever two strong nodes meet, they average values

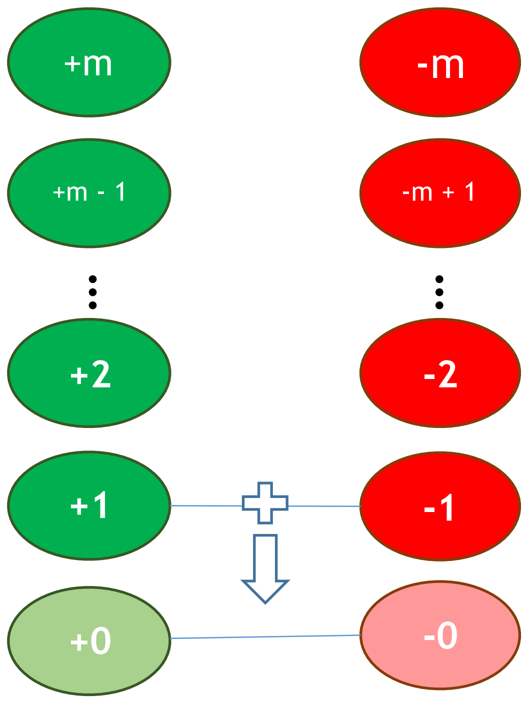


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

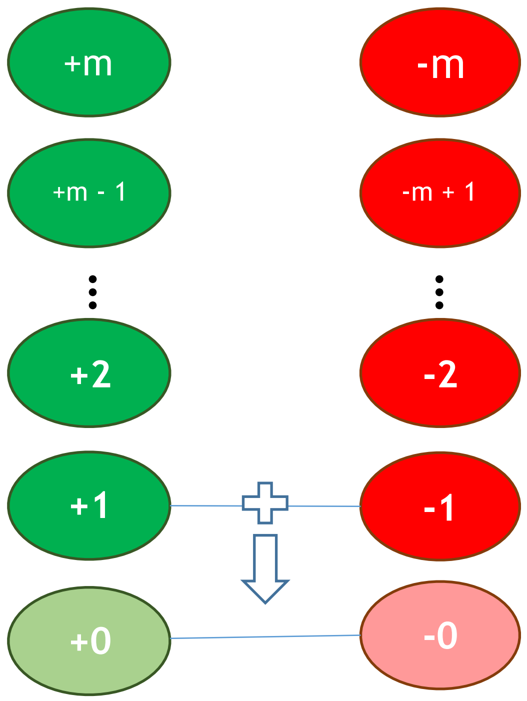


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

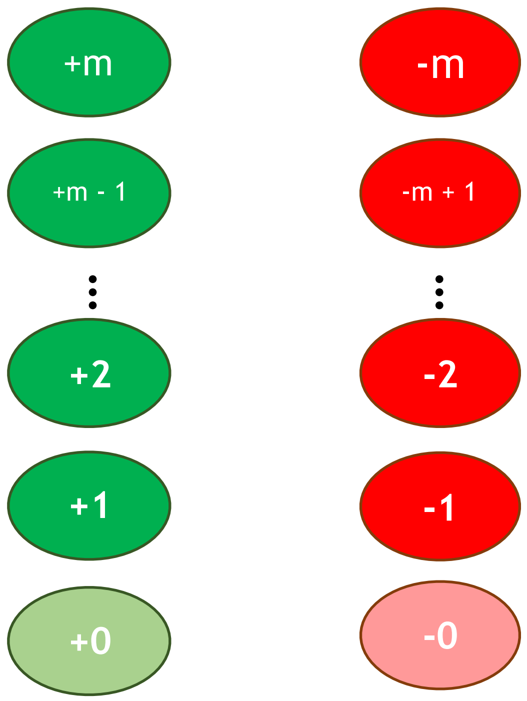


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

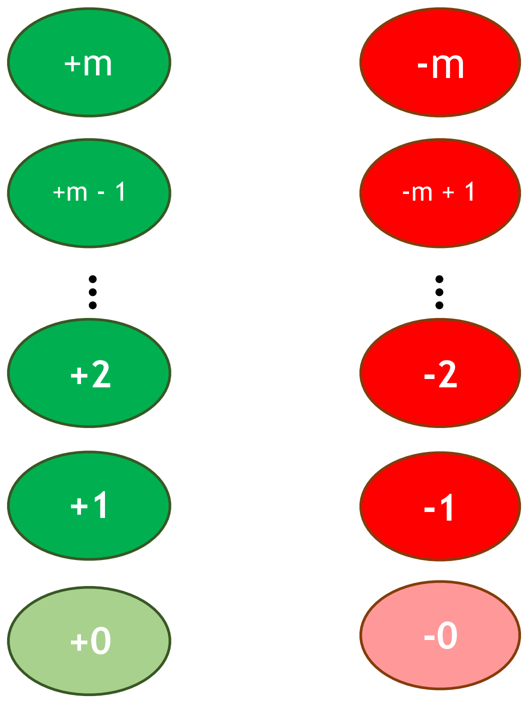


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:
- Whenever two strong nodes meet, they average values

Conquer:
- **Strong** nodes sway **weak** nodes towards their decision.

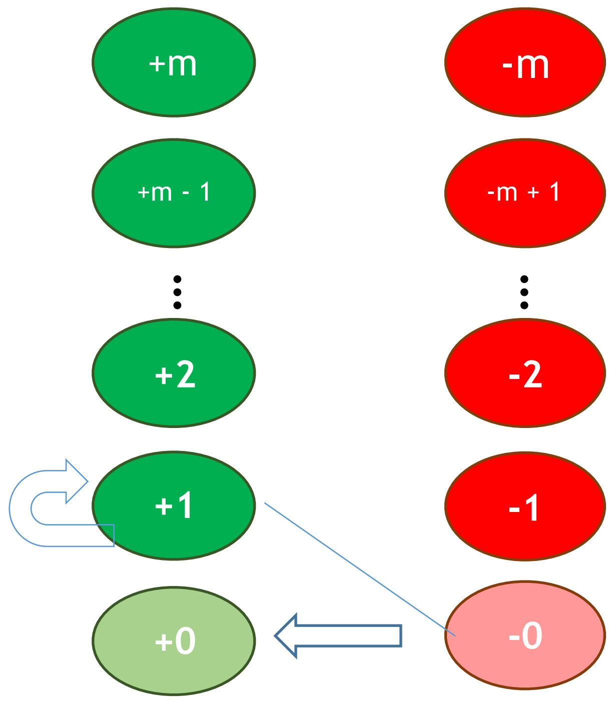


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:
- Whenever two strong nodes meet, they average values

Conquer:
- **Strong** nodes sway **weak** nodes towards their decision.

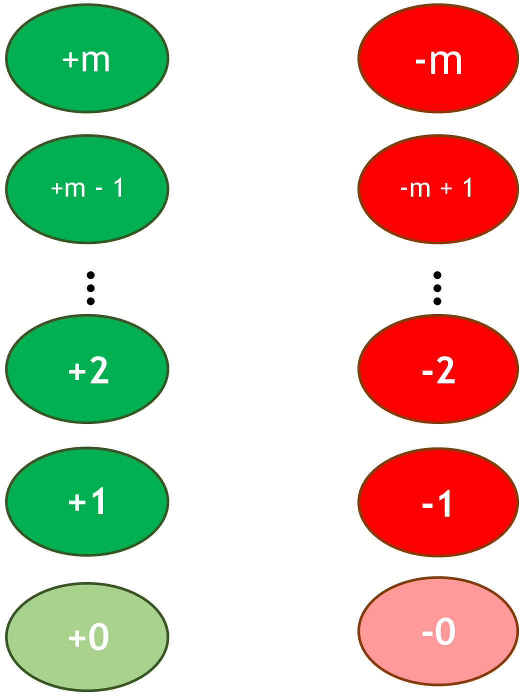


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:
- Whenever two strong nodes meet, they average values

Conquer:
- **Strong** nodes sway **weak** nodes towards their decision.

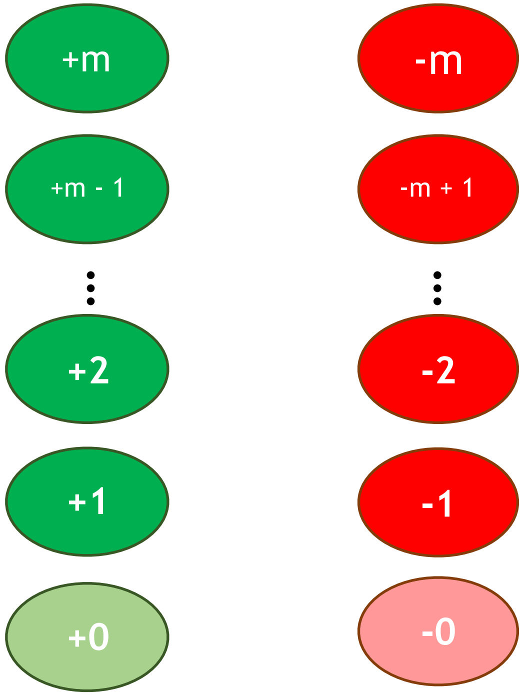


AVC in Action

Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:
- Whenever two strong nodes meet, they average values

Conquer:
- **Strong** nodes sway **weak** nodes towards their decision.



Note: For m = 1, we obtain a variant of 4EM.

AVC in Action

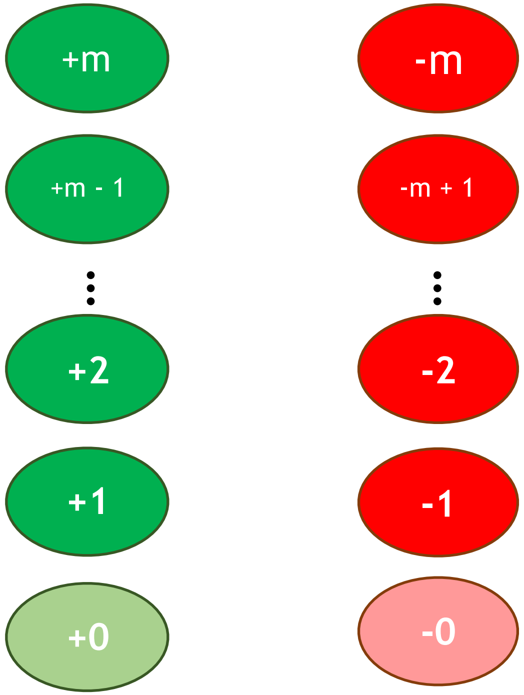
Initially: **+m** or **-m**, **odd** integers
Strong states: non-zero absolute value.
Weak states: value zero (+/-).

Averaging:

- Whenever two strong nodes meet, they average values

Conquer:

- **Strong** nodes sway **weak** nodes towards their decision.



Note: For m = 1, we obtain a variant of 4EM.

Disclaimer: original protocol is more complicated for technical reasons

Summing up

Summing up

Theorem 1 [AGV15]: Given fixed $m < n$, AVC solves majority exactly in expected parallel time $O(\log n / (m\,\varepsilon) + \log n \log m)$, using $s = O(m + \log n \log m)$ total states.

Summing up

Theorem 1 [AGV15]: Given fixed $m < n$, AVC solves majority exactly in expected parallel time $O(\log n / (m\,\varepsilon) + \log n \log m)$, using $s = O(m + \log n \log m)$ total states.

- In short:
  - If $m \approx 1 / \varepsilon$, then running time is always **poly-logarithmic**
  - If $\varepsilon = 1 / n$, then m needs to be **linear** in n
  - 10^{23} molecules -> $O(10^{23})$ states?!
  - 10^{23} molecules -> $O(23^2$ states $)$
  - The idea: ***quantize*** integer states to powers of two

Summing up

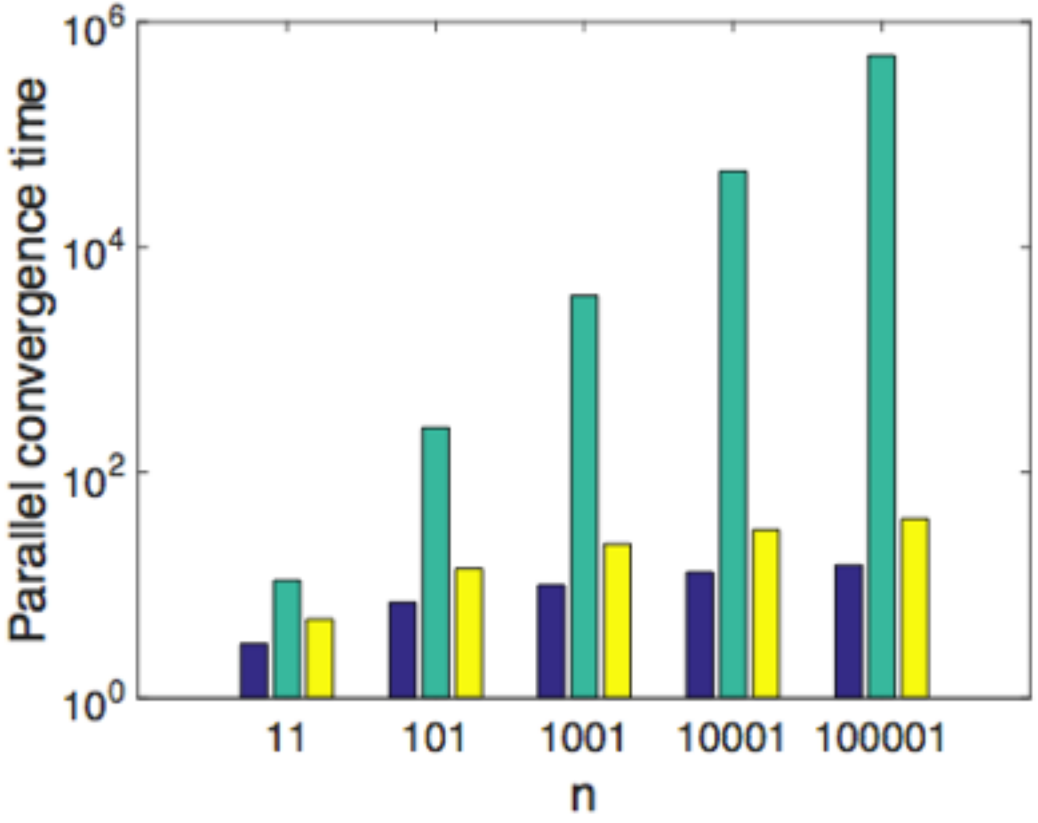
Theorem 1 [AGV15]: Given fixed $m < n$, AVC solves majority exactly in expected parallel time $O(\log n / (m\,\varepsilon) + \log n \log m)$, using $s = O(m + \log n \log m)$ total states.

- In short:
  - If **$m \approx 1 / \varepsilon$**, then running time is always **poly-logarithmic**
  - If **$\varepsilon = 1 / n$**, then m needs to be **linear** in n
  - 10^{23} molecules -> $O(10^{23})$ states?!

Theorem 2 [AAEGR16]: **logAVC** solves majority exactly in expected parallel time $O(\log^3 n)$, using $s = O(\log^2 n)$ total states.

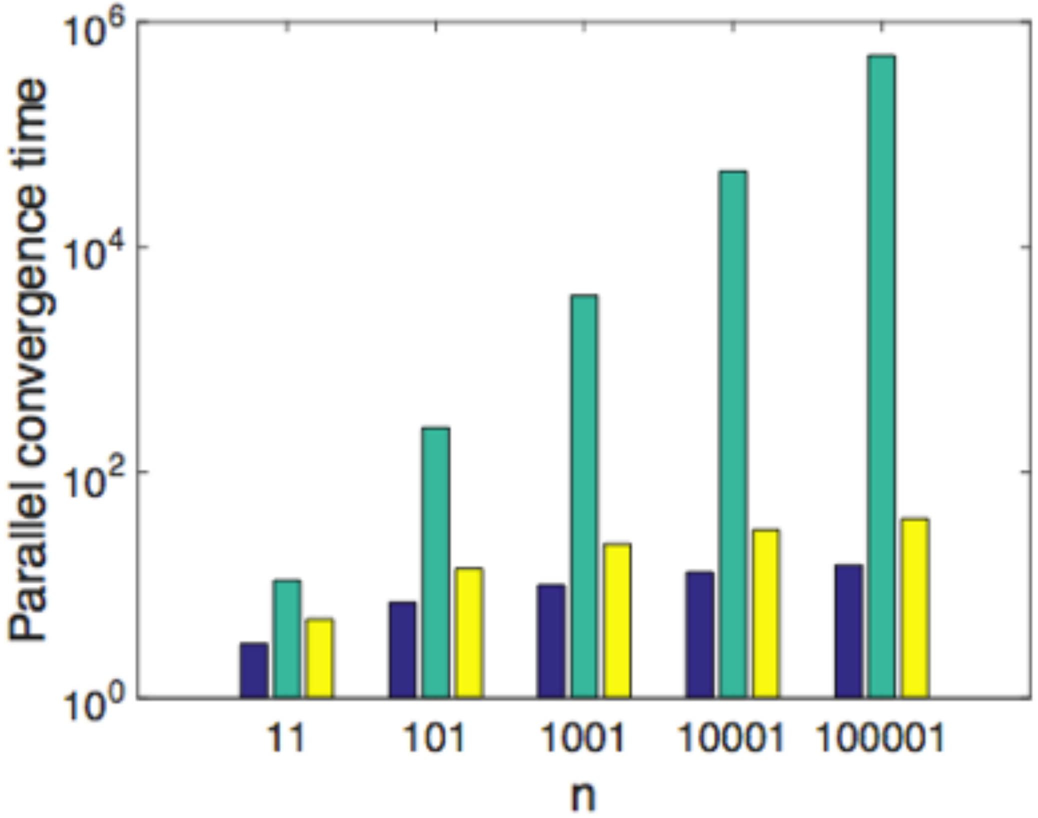
- 10^{23} molecules -> $O(23^2$ states)
- The idea: ***quantize*** integer states to powers of two

Is AVC any good?



Results are for **ε = O(1 / n)**
Legend:
Blue = 3AM
Green = 4EM
Yellow = AVC / logAVC

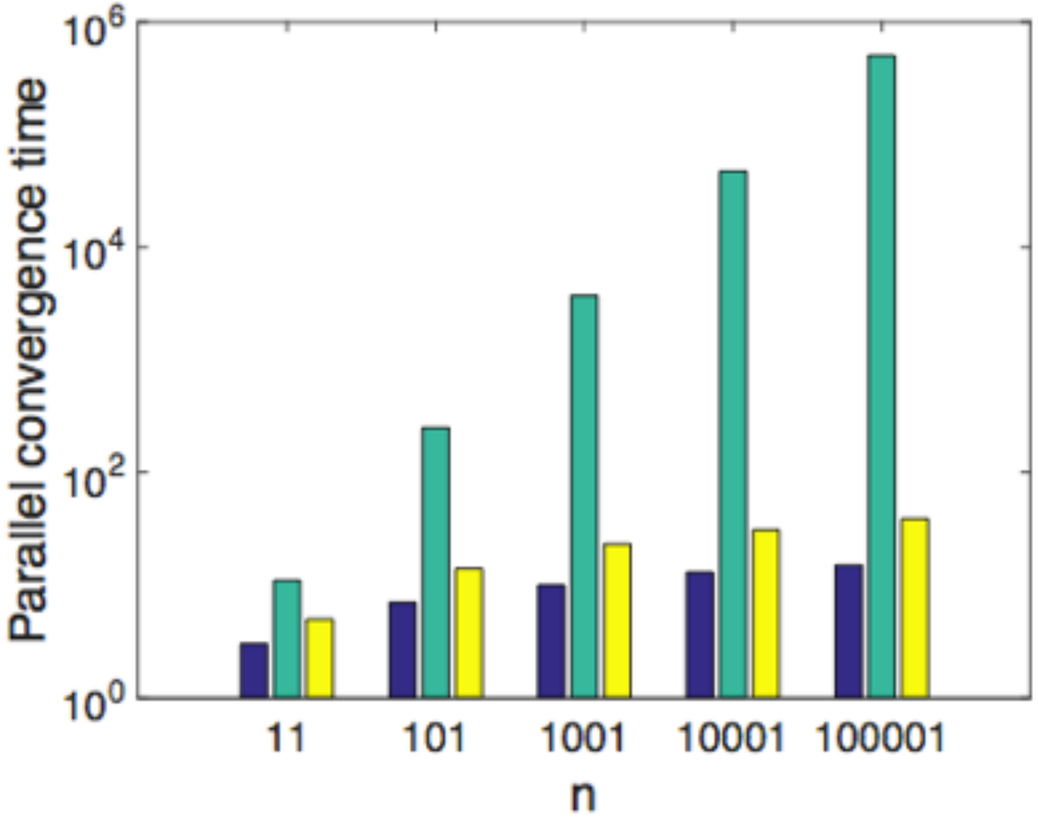
Is AVC any good?



Results are for **ε = O(1 / n)**
Legend:
Blue = 3AM
Green = 4EM
Yellow = AVC / logAVC

Is AVC implementable?

Is AVC any good?



Results are for **ε = O(1 / n)**
Legend:
Blue = 3AM
Green = 4EM
Yellow = AVC / logAVC

Is AVC implementable?

Challenging: currently, small constant number of states implementable.

Time-Space Trade-Offs

Time-Space Trade-Offs

Theorem A : Any protocol using $s < \frac{1}{2} \log \log n$ states per node and solving majority with **discrepancy ε** must have expected stabilization time $> n / (2^s + \varepsilon n)^2$.

Time-Space Trade-Offs

Theorem A : Any protocol using $s < \frac{1}{2} \log \log n$ states per node and solving majority with **discrepancy** ε must have expected stabilization time $> n / (2^s + \varepsilon n)^2$.

- In particular:
  - If **s = constant** and **εn = constant,** then stabilization time **linear in n**
  - If **s = O(loglog n)** and **εn = constant**, then stabilization time > **n / polylog n**

Time-Space Trade-Offs

Theorem A : Any protocol using $s < \frac{1}{2} \log \log n$ states per node and solving majority with discrepancy ε must have expected stabilization time $> n / (2^s + \varepsilon n)^2$.

- In particular:
  - If **s = constant** and **εn = constant**, then stabilization time **linear in n**
  - If **$s = O(\text{ loglog } n\text{ })$** and **$\varepsilon n$ = constant**, then stabilization time > **n / polylog n**

Complex molecules are needed for deterministic computation.

Discussion

Discussion

Molecular computation is fertile ground for algorithmic research.

Discussion

Molecular computation is fertile ground
for algorithmic research.

There are inherent space-time trade-offs when designing
deterministic population protocols.

Discussion

Molecular computation is fertile ground for algorithmic research.

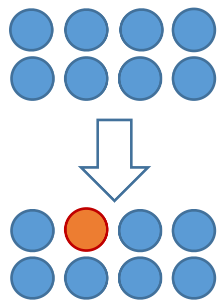
There are **inherent space-time trade-offs** when designing **deterministic** population protocols.

Open Challenges:

- Tighter trade-off bounds
- Other problems: **plurality**, **approximate counting**
- **Modeling faulty interactions** (leaks)
- **Large-scale simulation** of molecular algorithms

Leader Election

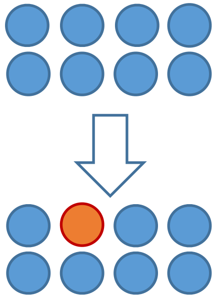
- **Input:** All nodes start in the same initial state
- **Output:**



Algorithm	Number of States	Convergence Time
Trivial Leader Election	2	$\Omega(n^2)$
Leader-Minion [AG, ICALP 2015]	$O(\log^3 n)$	$O(\log^3 n)$
Lottery Leader Election [AAEGR16]	$O(\log^2 n)$	$O(\log^{5.3} n \log\log n)$

Leader Election

- **Input:** All nodes start in the same initial state
- **Output:**
  - Exactly one node is in a “leader” state, remains leader forever



Algorithm	Number of States	Convergence Time
Trivial Leader Election	2	$\Omega(n^2)$
Leader-Minion [AG, ICALP 2015]	$O(\log^3 n)$	$O(\log^3 n)$
Lottery Leader Election [AAEGR16]	$O(\log^2 n)$	$O(\log^{5.3} n \log\log n)$

The Impossibility Result

Theorem A : Any protocol using $< \frac{1}{2} \log \log n$ states per node and electing L leaders will have expected stabilization time $> n / (C \text{ polylog } n \, L^2)$.

The Impossibility Result

Theorem A : Any protocol using $< \frac{1}{2} \log \log n$ states per node and electing L leaders will have expected stabilization time $> n / (C \text{ polylog } n\, L^2)$.

- Example:
  - **O(log log n) states / node**, **one** leader
  - Stabilization time > n / polylog n (quasi-linear)
  - Generalizes a recent result by Doty and Soloveichik [DISC15] to super-constant states

Bonus: A Cute Algorithm

- The goal: approximate **n**
- The state:
  - A flip bit F, initially 0
  - A counter “variable” C, initially 0
- The algorithm:
  - Stage 1: do four interactions, updating $F = 1 - F'$
  - Stage 2: increment counter C until you first see $F' = 1$
  - Stage 3: exchange C with interaction partner, setting $C = \max(C, C')$
- The guarantee:
  - The convergence value is
    $(1 - \mathrm{eps}) \log n < C < (1 + \mathrm{eps}) \log n$,
    with high probability