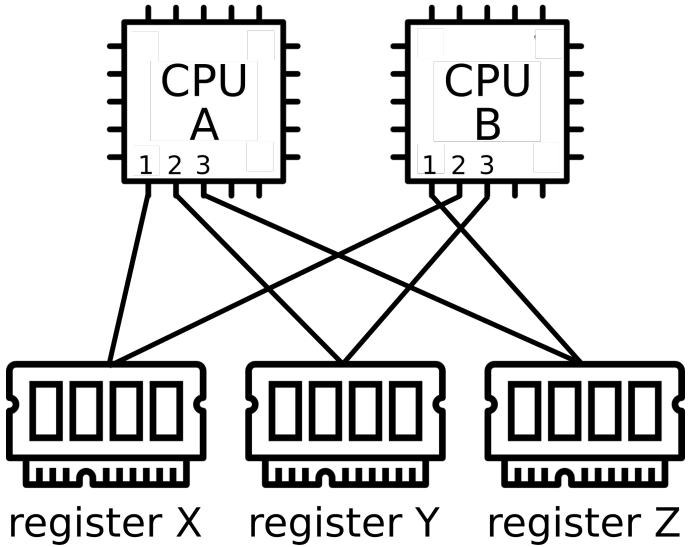
Brief Announcement: Understanding Read-Write Wait-Free Coverings in the Fully-Anonymous Shared-Memory Model

Giuliano Losa, Stellar Development Foundation

Eli Gafni, UCLA

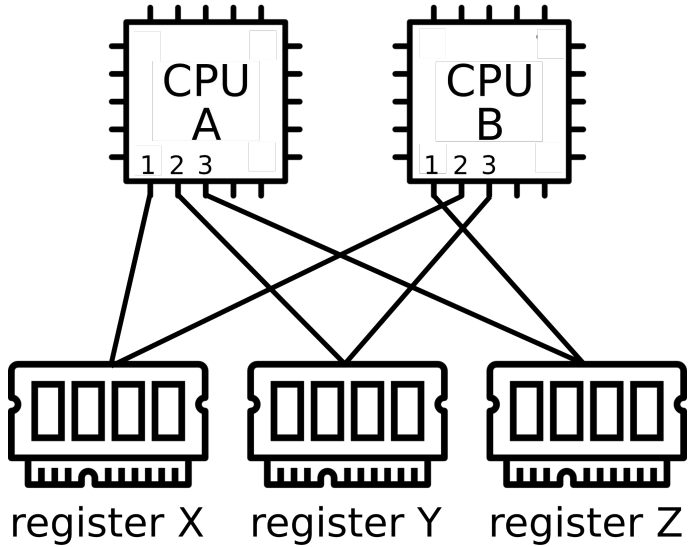
What can we compute wait-free in the fully-anonymous shared memory model?

- Processors are anonymous
  - All run the same program and do not know their ID
- Memory is anonymous:
  - Processors are wired to atomic read/write registers in an arbitrary way
  - Processors do not know how they are wired



What can we compute wait-free in the fully-anonymous shared memory model?

- Processors are anonymous
  - All run the same program and do not know their ID
- Memory is anonymous:
  - Processors are wired to atomic read/write registers in an arbitrary way
  - Processors do not know how they are wired
- Main difficulty: not knowing where they write, processors can hardly avoid overwriting each other
  - E.g. obstruction-free consensus is impossible with less than N registers



Why? We think it is interesting to investigate what remains computable when we remove common assumptions

There are at least a few other authors and reviewers that (presumably) think so too:

- Gadi Taubenfeld. *Anonymous shared memory*, JACM, 2022
- Raynal and Taubenfeld. *Fully anonymous consensus and set agreement algorithms*, 2021
- Raynal and Taubenfeld. *Mutual exclusion in fully anonymous shared memory systems*
- Imbs, Raynal, Taubenfeld, and Parter. *Election in Fully Anonymous Shared Memory Systems: Tight Space Bounds and Algorithms*, SICC, 2022
- Aghazadeh, Imbs, Raynal, Taubenfeld, Woelfel. *Optimal Memory-Anonymous Symmetric Deadlock-Free Mutual Exclusion*, PODC, 2019
- Godard, Imbs, Raynal, Taubenfeld. *From Bezout's Identity to Space-Optimal Election in Anonymous Memory Systems*, PODC, 2020

Consider this simple program:

```
i := 1
view := {input}
while true:
    read all the registers
    view := view ∪ (values read)
    write view to port i
    i := i+1
```

Despite reading and writing forever, there are executions where some processors never see each other

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	$\{\}$	$\underline{\{1\}}$	$\underline{\{1\}}$	$\{1\}$	$\{2\}$	$\{3\}$
2	p_2 writes then scans	$\{2\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\underline{\{1,2\}}$	$\{3\}$
3	p_3 overwrites p_2 then scans	$\underline{\{3\}}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\underline{\{1,3\}}$

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	$\underline{\{1\}}$	$\underline{\{1\}}$	{1}	{2}	{3}
2	p_2 writes then scans	$\underline{\{2\}}$	{1}	{1}	{1}	$\underline{\{1,2\}}$	{3}
3	p_3 overwrites p_2 then scans	$\underline{\{3\}}$	{1}	{1}	{1}	{1,2}	$\underline{\{1,3\}}$
4	p_1 overwrites p_3 then scans	$\underline{\{1\}}$	{1}	{1}	{1}	{1,2}	{1,3}
5	p_2 writes then scans	{1}	{1,2}	{1}	{1}	{1,2}	{1,3}
6	p_3 overwrites p_2 then scans	{1}	$\underline{\{1,3\}}$	{1}	{1}	{1,2}	{1,3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	$\underline{\{1\}}$	$\underline{\{1\}}$	{1}	{2}	{3}
2	p_2 writes then scans	$\underline{\{2\}}$	{1}	{1}	{1}	$\underline{\{1,2\}}$	{3}
3	p_3 overwrites p_2 then scans	$\underline{\{3\}}$	{1}	{1}	{1}	{1,2}	$\underline{\{1,3\}}$
4	p_1 overwrites p_3 then scans	$\underline{\{1\}}$	{1}	{1}	{1}	{1,2}	{1,3}
5	p_2 writes then scans	{1}	$\underline{\{1,2\}}$	{1}	{1}	{1,2}	{1,3}
6	p_3 overwrites p_2 then scans	{1}	$\underline{\{1,3\}}$	{1}	{1}	{1,2}	{1,3}
7	p_1 overwrites p_3 then scans	{1}	$\underline{\{1\}}$	{1}	{1}	{1,2}	{1,3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}
6	p_3 overwrites p_2 then scans	{1}	{1, 3}	{1}	{1}	{1, 2}	{1, 3}
7	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
8	p_2 writes then scans	{1}	{1}	{1, 2}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}
6	p_3 overwrites p_2 then scans	{1}	{1, 3}	{1}	{1}	{1, 2}	{1, 3}
7	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
8	p_2 writes then scans	{1}	{1}	{1, 2}	{1}	{1, 2}	{1, 3}
9	p_3 overwrites p_2 then scans	{1}	{1}	{1, 3}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}
6	p_3 overwrites p_2 then scans	{1}	{1, 3}	{1}	{1}	{1, 2}	{1, 3}
7	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
8	p_2 writes then scans	{1}	{1}	{1, 2}	{1}	{1, 2}	{1, 3}
9	p_3 overwrites p_2 then scans	{1}	{1}	{1, 3}	{1}	{1, 2}	{1, 3}
10	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}
6	p_3 overwrites p_2 then scans	{1}	{1, 3}	{1}	{1}	{1, 2}	{1, 3}
7	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
8	p_2 writes then scans	{1}	{1}	{1, 2}	{1}	{1, 2}	{1, 3}
9	p_3 overwrites p_2 then scans	{1}	{1}	{1, 3}	{1}	{1, 2}	{1, 3}
10	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
11	p_2 writes then scans	{1, 2}	{1}	{1}	{1}	{1, 2}	{1, 3}

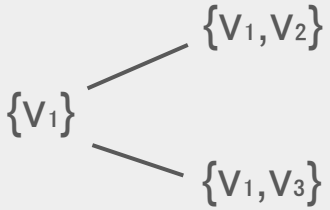
	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	$\underline{\{1\}}$	$\underline{\{1\}}$	{1}	{2}	{3}
2	p_2 writes then scans	$\underline{\{2\}}$	{1}	{1}	{1}	$\underline{\{1,2\}}$	{3}
3	p_3 overwrites p_2 then scans	$\underline{\{3\}}$	{1}	{1}	{1}	{1,2}	$\underline{\{1,3\}}$
4	p_1 overwrites p_3 then scans	$\underline{\{1\}}$	{1}	{1}	{1}	{1,2}	{1,3}
5	p_2 writes then scans	{1}	$\underline{\{1,2\}}$	{1}	{1}	{1,2}	{1,3}
6	p_3 overwrites p_2 then scans	{1}	$\underline{\{1,3\}}$	{1}	{1}	{1,2}	{1,3}
7	p_1 overwrites p_3 then scans	{1}	$\underline{\{1\}}$	{1}	{1}	{1,2}	{1,3}
8	p_2 writes then scans	{1}	{1}	$\underline{\{1,2\}}$	{1}	{1,2}	{1,3}
9	p_3 overwrites p_2 then scans	{1}	{1}	$\underline{\{1,3\}}$	{1}	{1,2}	{1,3}
10	p_1 overwrites p_3 then scans	{1}	{1}	$\underline{\{1\}}$	{1}	{1,2}	{1,3}
11	p_2 writes then scans	{1,2}	{1}	{1}	{1}	{1,2}	{1,3}
12	p_3 overwrites p_2 then scans	$\underline{\{1,3\}}$	{1}	{1}	{1}	{1,2}	{1,3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}
6	p_3 overwrites p_2 then scans	{1}	{1, 3}	{1}	{1}	{1, 2}	{1, 3}
7	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
8	p_2 writes then scans	{1}	{1}	{1, 2}	{1}	{1, 2}	{1, 3}
9	p_3 overwrites p_2 then scans	{1}	{1}	{1, 3}	{1}	{1, 2}	{1, 3}
10	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
11	p_2 writes then scans	{1, 2}	{1}	{1}	{1}	{1, 2}	{1, 3}
12	p_3 overwrites p_2 then scans	{1, 3}	{1}	{1}	{1}	{1, 2}	{1, 3}
13	p_1 overwrites p_3 then scans (same as 4)	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and ends with a scan	{}	{1}	{1}	{1}	{2}	{3}
2	p_2 writes then scans	{2}	{1}	{1}	{1}	{1, 2}	{3}
3	p_3 overwrites p_2 then scans	{3}	{1}	{1}	{1}	{1, 2}	{1, 3}
4	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
5	p_2 writes then scans	{1}	{1, 2}	{1}	{1}	{1, 2}	{1, 3}
6	p_3 overwrites p_2 then scans	{1}	{1, 3}	{1}	{1}	{1, 2}	{1, 3}
7	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
8	p_2 writes then scans	{1}	{1}	{1, 2}	{1}	{1, 2}	{1, 3}
9	p_3 overwrites p_2 then scans	{1}	{1}	{1, 3}	{1}	{1, 2}	{1, 3}
10	p_1 overwrites p_3 then scans	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}
11	p_2 writes then scans	{1, 2}	{1}	{1}	{1}	{1, 2}	{1, 3}
12	p_3 overwrites p_2 then scans	{1, 3}	{1}	{1}	{1}	{1, 2}	{1, 3}
13	p_1 overwrites p_3 then scans (same as 4)	{1}	{1}	{1}	{1}	{1, 2}	{1, 3}

	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
	p_1 writes twice and						
	then scans						
11	p_2 writes then scans	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$
12	p_3 overwrites p_2 then scans	$\{1,3\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$
13	p_1 overwrites p_3 then scans (same as 4)	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$

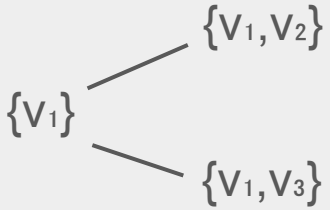
stable view graph

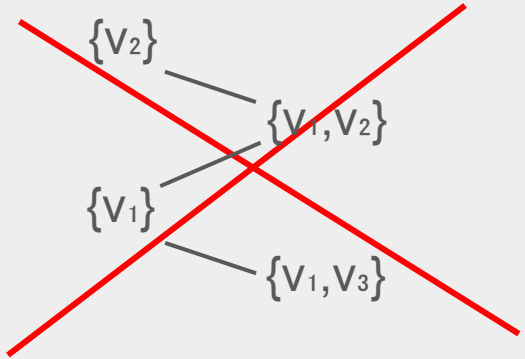


	Actions	Post State					
		r_1	r_2	r_3	$view[p_1]$	$view[p_2]$	$view[p_3]$
1	p_1 writes twice and						
	then scans						
11	p_2 writes then scans	$\{1,2\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$
12	p_3 overwrites p_2 then scans	$\{1,3\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$
13	p_1 overwrites p_3 then scans (same as 4)	$\{1\}$	$\{1\}$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$

We prove that there always is a minimum stable view

stable view graph





Contributions

1. How to make sense of colored tasks like renaming in an anonymous model? We propose using group-solvability (Gafni 2004)
2. Stable-view graphs have a unique source
3. We solve the snapshot task (a.k.a. lattice agreement) wait-free using N registers
4. Using snapshots, we solve wait-free (group) renaming and obstruction-free consensus using N register

Open question

Characterize the set of tasks that are (group-)solvable under full anonymity