OS Synchronization as Symmetry Restrictions in S_n

A Computational and Structural Study

Om Pranab Mohanty

2026

Abstract

Classical operating system synchronization mechanisms — mutual exclusion, round-robin scheduling, and deadlock — are typically described in operational terms: semaphores, locks, and wait queues. This paper proposes a complementary structural description using elementary group theory. We model the space of all process schedules for a single scheduling epoch as the symmetric group S_n, where each permutation represents a total ordering of n concurrent processes under the assumptions of exactly one execution slot per epoch, no preemption, and no priority.

We show that three synchronization constraints correspond to well-known algebraic substructures: mutual exclusion corresponds to a stabilizer subgroup $M(x) = \mathrm{Stab}_{S_n}(x) \leq S_n$ of order $(n-1)!$; round-robin scheduling corresponds to a cyclic subgroup $\langle c \rangle \cong \mathbb{Z}_n$ of order n; and deadlock, under an explicit modeling assumption, corresponds to the identity element $e \in S_n$.

The subgroups $M(x)$ and $\langle c \rangle$ are incomparable for $n \geq 3$, with $M(x) \cap \langle c \rangle = \{e\}$, reflecting a structural separation between safety and fairness constraints in the scheduling space. These correspondences are verified computationally in Python for $n \in \{2, 3, 4, 5, 6\}$ using only first-principles implementations with no computer algebra library. Source code and notebooks are available at [5].

Contents

1 Introduction

Operating systems manage concurrent processes through synchronization primitives. These primitives are well understood operationally: a mutex prevents two processes from entering a critical section simultaneously; a round-robin scheduler grants each process a fair time quantum in cyclic order; a deadlock is a state in which a circular chain of waiting processes prevents any forward progress [1].

What is less often examined is whether these mechanisms admit a common *structural* description. This paper proposes one, grounded in permutation group theory [2]. The central observation is that any total ordering of n concurrent processes is a bijection on the process set — a permutation — and the set of all such orderings is the symmetric group S_n. Synchronization constraints restrict which orderings are admissible. We show that for three classical constraints, the admissible sets are subgroups of S_n with recognizable algebraic structure.

Not every synchronization constraint defines a subgroup of S_n. We focus specifically on constraints whose admissible sets are closed under composition and inverse, verifying this closure formally and computationally for each case.

Contribution

This paper makes the following conceptual contributions. First, it identifies synchronization constraints as subgroup restrictions on S_n, reframing operational rules as structural conditions on the scheduling space. Second, it observes that safety (mutual exclusion) and fairness (round-robin) correspond to incomparable subgroups — $M(x)$ and $\langle c \rangle$ meet only at $\{e\}$ for $n \geq 3$ — which gives a structural account of why these constraints are in tension. Third, it provides an executable verification of this picture implemented from first principles [5]. The underlying group-theoretic results are standard; the contribution lies in the unification and interpretation.

Related Work

To the best of the author's knowledge, the explicit use of subgroup lattices to classify OS synchronization primitives has not been developed in the standard literature. Standard OS texts [1] treat synchronization operationally. Algebraic approaches to concurrency exist (e.g., process algebras such as CSP and CCS), but these use operational rather than structural algebraic models and do not employ permutation group theory. The use of group actions in theoretical computer science is well-established in areas such as Burnside counting and symmetry reduction in model checking, but the specific mapping developed here — scheduling epochs as elements of S_n, constraints as subgroups — does not appear to have been made explicit in the accessible literature.

Scope

This is a formalization and verification project. The model covers total orderings within a single scheduling epoch and does not capture preemption, priority scheduling, or partial orders. The deadlock claim relies on an explicit modeling assumption stated in Section 8.

2 Model Assumptions

All claims in Sections 5–8 are made within the following explicitly stated modeling boundary.

Assumption 1 (Finite Process Set). The system contains exactly $n \geq 2$ distinguishable processes, labelled $1, 2, \ldots, n$. We write $P = \{1, \ldots, n\}$.

Assumption 2 (Single Scheduling Epoch). A schedule represents a single epoch of execution — a complete assignment of all n processes to n execution slots at one point in time. The model does not capture dynamic time evolution, preemption, context switching between epochs, or multi-step scheduling histories.

Assumption 3 (Total Ordering). Each epoch assigns each process exactly one execution slot, and each slot is assigned to exactly one process. A schedule is therefore a bijection on P: no two processes share a slot, and no process is unscheduled.

Assumption 4 (No Preemption Within an Epoch). Once an epoch's schedule is fixed, it executes to completion without interruption or rescheduling.

Assumption 5 (No Priority or Probabilistic Weighting). All processes are treated as equally schedulable. The model contains no weights, priority levels, or probability distributions over schedules.

Assumption 6 (Single Critical Slot). The mutual exclusion constraint is modelled with respect to a single designated slot $x \in P$. Generalisation to multi-resource mutex via intersections of stabilizers is noted in Section 11.4 but not developed here.

3 Background

3.1 Symmetric Groups

We recall the necessary definitions. The primary algebraic reference is Gallian [2].

Definition 3.1 (Symmetric Group). Let $P = \{1, 2, \ldots, n\}$. The *symmetric group* S_n is the set of all bijections $\sigma : P \to P$, equipped with function composition:

$$(\sigma \circ \tau)(i) = \sigma(\tau(i)), \quad i \in P.$$

Proposition 3.2. *$(S_n, \circ)$ is a group of order $|S_n| = n!$*

Definition 3.3 (Subgroup). A non-empty subset $H \subseteq S_n$ is a *subgroup*, written $H \leq S_n$, if it contains the identity element, is closed under composition, and is closed under inverses.

Theorem 3.4 (Lagrange [2]). *If $H \leq S_n$, then $|H|$ divides $|S_n| = n!$*

Definition 3.5 (Group Action, Orbit, Stabilizer). The natural *action* of S_n on P is $(\sigma, i) \mapsto \sigma(i)$. For $x \in P$:

$$\mathrm{Orb}(x) = \{\sigma(x) \mid \sigma \in S_n\}, \qquad \mathrm{Stab}(x) = \{\sigma \in S_n \mid \sigma(x) = x\}.$$

Theorem 3.6 (Orbit-Stabilizer [2]). *For any $x \in P$: $|S_n| = |\mathrm{Orb}(x)| \cdot |\mathrm{Stab}(x)|$.*

3.2 Cycles and Cyclic Subgroups

Definition 3.7 (Cycle Notation)**.** The permutation $(a_1\ a_2\ \cdots\ a_k)$ maps $a_1 \mapsto a_2 \mapsto \cdots \mapsto a_k \mapsto a_1$ and fixes all other elements. Such a permutation is called a *k-cycle*.

Definition 3.8 (Cyclic Subgroup)**.** For $\sigma \in S_n$, the *cyclic subgroup* generated by σ is $\langle\sigma\rangle = \{\sigma^k \mid k \in \mathbb{Z}\}$, where $\sigma^0 = e$.

Lemma 3.9 (Order of an n-Cycle)**.** *Any n-cycle in S_n has order n.*

Proof. Let $c = (a_1\ a_2\ \cdots\ a_n)$. Then $c^k(a_1) = a_{1+k \bmod n}$. For $c^k = e$ we need $1 + k \equiv 1 \pmod{n}$, i.e. $n \mid k$. The smallest positive such k is n, so $\mathrm{ord}(c) = n$. □

4 Formal Model

Definition 4.1 (Schedule)**.** Under Assumptions 1–3, a *schedule* is a bijection $\sigma : P \to P$. If $\sigma(i) = k$, process i is assigned to execution slot k in the current epoch.

Remark 4.2. A schedule is a static assignment for one epoch. It does not model the dynamic process by which schedules are selected over time, nor properties such as liveness or starvation across multiple epochs.

Definition 4.3 (Scheduling Space)**.** The *unrestricted scheduling space* is S_n — the set of all schedules admitted by Assumptions 1–3.

Definition 4.4 (Synchronization Constraint)**.** A *synchronization constraint* is a predicate $C : S_n \to \{0, 1\}$. The *admissible set* is $\mathcal{A}_C = \{\sigma \in S_n \mid C(\sigma) = 1\}$.

Remark 4.5 (Subgroup Requirement). Not every synchronization constraint produces a subgroup. For each constraint studied here, we verify explicitly that $\mathcal{A}_C$ is closed under composition, closed under inverses, and contains the identity. This makes $\mathcal{A}_C \leq S_n$. Closure is proved in each case, not assumed.

5 Claim 1: The Scheduling Space is S_n

Proposition 5.1. *The unrestricted scheduling space has order $|S_n| = n!$*

Proof. A bijection from P to P is determined by assigning images to $1, 2, \ldots, n$ sequentially: n choices for $\sigma(1)$, $n-1$ for $\sigma(2)$, and so on. The total is $n \cdot (n-1) \cdots 1 = n!$. □

n	$\|S_n\|$	$n!$
2	2	2
3	6	6
4	24	24
5	120	120
6	720	720

Table 1: Size of the scheduling space S_n, verified computationally.

6 Claim 2: Mutual Exclusion as a Stabilizer Subgroup

6.1 The Mutex Constraint

We model the critical section as a designated slot $x \in P$ under Assumption 6. A schedule respects mutual exclusion if the process assigned to slot x is not displaced:

Definition 6.1 (Mutex-Admissible Schedule)**.** $\sigma \in S_n$ is *mutex-admissible* with respect to slot $x \in P$ if $\sigma(x) = x$.

Theorem 6.2. *Let $M(x) = \{\sigma \in S_n \mid \sigma(x) = x\}$. Then:*

1. $M(x) = \mathrm{Stab}_{S_n}(x)$ *is a subgroup of* S_n.
2. $|M(x)| = (n-1)!$
3. $M(x) \cong S_{n-1}$.
4. *The index* $[S_n : M(x)] = n$.

Proof. (i) Subgroup. $e(x) = x$, so $e \in M(x)$. For $\sigma, \tau \in M(x)$: $(\sigma \circ \tau)(x) = \sigma(\tau(x)) = \sigma(x) = x$, so $M(x)$ is closed under composition. If $\sigma(x) = x$, applying σ^{-1} gives $x = \sigma^{-1}(x)$, so $M(x)$ is closed under inverses. Hence $M(x) \leq S_n$.

(ii) Order. The orbit of any $x \in P$ under S_n is all of P (for any $y \in P$, there exists σ mapping x to y), so $|\mathrm{Orb}(x)| = n$. By Theorem 3.6: $n! = n \cdot |M(x)|$, giving $|M(x)| = (n-1)!$

(iii) Isomorphism. Any $\sigma \in M(x)$ fixes x and acts freely on $P \setminus \{x\}$. The restriction map $\sigma \mapsto \sigma|_{P\setminus\{x\}}$ is a group isomorphism $M(x) \to S_{n-1}$.

(iv) Index. $[S_n : M(x)] = n!/(n-1)! = n$. □

6.2 Coset Decomposition

The n cosets of $M(x)$ in S_n partition the scheduling space. The identity coset $M(x)$ contains all admissible schedules. Each of the remaining $n-1$ cosets corresponds to a distinct process illegally occupying slot x:

$$S_n = M(x) \;\cup\; \sigma_1 M(x) \;\cup\; \cdots \;\cup\; \sigma_{n-1} M(x).$$

n	$\lvert S_n\rvert$	$\lvert M(x)\rvert$	$(n-1)!$	Index	Subgroup
2	2	1	1	2	✓
3	6	2	2	3	✓
4	24	6	6	4	✓
5	120	24	24	5	✓
6	720	120	120	6	✓

Table 2: Stabilizer subgroup orders, verified computationally.

7 Claim 3: Round-Robin as a Cyclic Subgroup

Definition 7.1 (Round-Robin Permutation)**.** The *round-robin permutation* is the n-cycle:

$$c = (1\ 2\ 3\ \cdots\ n), \qquad c(i) = \begin{cases} i+1 & i < n, \\ 1 & i = n. \end{cases}$$

Theorem 7.2. *The cyclic subgroup $\langle c \rangle \leq S_n$ satisfies:*

1. $|\langle c \rangle| = n$.
2. $\langle c \rangle \cong \mathbb{Z}_n$.
3. $\langle c \rangle$ *acts transitively on* P.
4. $\langle c \rangle$ *is abelian.*
5. $\langle c \rangle$ *is not normal in* S_n *for* $n \geq 3$.

Proof. (i) By Lemma 3.9, $\mathrm{ord}(c) = n$, so $|\langle c \rangle| = n$.

(ii) The map $\varphi : \mathbb{Z}_n \to \langle c \rangle$, $\varphi(k) = c^k$, is a bijective homomorphism: $\varphi(j+k \bmod n) = c^{j+k} = c^j \circ c^k = \varphi(j) \circ \varphi(k)$.

(iii) For any $i, j \in P$, set $k = j - i \bmod n$. Then $c^k(i) = i + k \equiv j \pmod{n}$, so every process reaches every slot. The action is transitive.

(iv) $c^j \circ c^k = c^{j+k} = c^k \circ c^j$ since integer addition commutes.

(v) For $n \geq 3$, let $\tau = (1\ 2) \in S_n$. Then $\tau c \tau^{-1} = (2\ 1\ 3\ 4\ \cdots\ n)$, which maps $2 \to 1$ rather than $2 \to 3$. This permutation is not a power of c, so $\tau \langle c \rangle \tau^{-1} \neq \langle c \rangle$, and $\langle c \rangle$ is not normal in S_n. □

Remark 7.3. Points (iii) and (v) deserve OS interpretation. Transitivity means every process will eventually be scheduled regardless of the starting state — this is starvation freedom. Non-normality for $n \geq 3$ means the round-robin subgroup does not admit a well-defined quotient with the full scheduling space, distinguishing it from the alternating group $A_n \trianglelefteq S_n$.

n	$\|\langle c \rangle\|$	$\cong \mathbb{Z}_n$	Transitive	Abelian	Normal in S_n
2	2	✓	✓	✓	✓
3	3	✓	✓	✓	×
4	4	✓	✓	✓	×
5	5	✓	✓	✓	×
6	6	✓	✓	✓	×

Table 3: Cyclic subgroup properties, verified computationally.

8 Claim 4: Deadlock as the Identity Element

8.1 Modeling Assumption

Assumption 7 (Deadlock as Identity)**.** In a deadlock state, no process can make forward progress. Within the epoch model, we represent this as: the schedule σ satisfies $\sigma(i) = i$ for all $i \in P$. No process is moved to a new execution slot.

Assumption 7 is an *operational identification*, not a derivation from the Coffman conditions [3]. The Coffman conditions — mutual exclusion, hold-and-wait, no preemption, circular wait — characterize the system state from which deadlock arises. We do not derive Assumption 7 from these conditions; we state it independently as a modeling choice appropriate to the static, epoch-based framework. The dynamic process by which a system arrives at deadlock is not modelled here.

The identity is the natural choice in this model for the following reason. A schedule $\sigma \in S_n$ is a bijection representing process advancement: $\sigma(i) = j$ means process i is moved to slot j. The only bijection under which no process is moved — the only element of S_n representing a complete absence of scheduling action — is the identity e. There is no zero element in S_n (groups have no absorbing element), so the identity is the unique candidate. One could alternatively model deadlock as a fixed-point subspace of a broader action, but within the bijection model of Assumption 3, the identity is not merely a convenient choice; it is the only element consistent with the requirement that every process maps to itself.

8.2 Formal Statement

Proposition 8.1. *Under Assumption 7, the deadlock state corresponds to the identity element $e \in S_n$. Moreover, e is the unique element of S_n satisfying $\sigma(i) = i$ for all $i \in P$.*

Proof. The identity function on a finite set is unique. Any σ satisfying $\sigma(i) = i$ for all i is by definition the identity. Existence of e in S_n is standard. □

8.3 Circular Wait versus Deadlock State

The Coffman circular-wait condition describes a cycle of blocking: p_1 waits for p_2, p_2 for p_3, …, p_k for p_1. Expressed in permutation terms, this dependency chain has the algebraic form of a k-cycle $(p_1\ p_2\ \cdots\ p_k)$ — identical in form to the round-robin generator c of Section 7.

The two objects live in different spaces and carry opposite meanings:

Object	Space	Interpretation
$(1\ 2\ \cdots\ n) \in \langle c \rangle$	Scheduling space S_n	Progress: processes run in cyclic order
$(p_1\ p_2\ \cdots\ p_k)$ in wait graph	Resource dependency graph	Circular wait: each process blocks the next

The algebraic form is identical; context determines meaning. The deadlock *end state* — when no admissible non-trivial scheduling action remains — is captured by e. The mechanism of arrival is not part of this model.

9 Worked Example: $n = 3$

We ground the four claims concretely for $n = 3$, $P = \{1, 2, 3\}$.

Example 9.1. The six elements of S_3 are:

$$S_3 = \{e,\ (1\ 2),\ (1\ 3),\ (2\ 3),\ (1\ 2\ 3),\ (1\ 3\ 2)\}$$

Claim 1. $|S_3| = 6 = 3!$ ✓

Claim 2. $M(1) = \{e,\ (2\ 3)\}$, $|M(1)| = 2 = (3-1)!$ ✓

Coset decomposition:

$$\begin{aligned} M(1) &= \{e,\ (2\ 3)\} && \text{admissible (slot 1 held by process 1)} \\ (1\ 2)\,M(1) &= \{(1\ 2),\ (1\ 2\ 3)\} && \text{violation (process 2 in slot 1)} \\ (1\ 3)\,M(1) &= \{(1\ 3),\ (1\ 3\ 2)\} && \text{violation (process 3 in slot 1)} \end{aligned}$$

Claim 3. $c = (1\ 2\ 3)$, $\langle c\rangle = \{e,\ (1\ 2\ 3),\ (1\ 3\ 2)\} \cong \mathbb{Z}_3$, $|\langle c\rangle| = 3$ ✓

Round-robin orbit: $1 \xrightarrow{c} 2 \xrightarrow{c} 3 \xrightarrow{c} 1$ (transitive) ✓

Claim 4. The unique element of S_3 satisfying $\sigma(i) = i$ for all i is e ✓

Incomparability. $M(1) \cap \langle c\rangle = \{e\}$, confirming $M(1)$ and $\langle c\rangle$ are incomparable for $n = 3$.

10 Computational Verification

10.1 Implementation

All claims are verified in Python without symbolic algebra libraries. Permutations are represented as dicts $\{i : \sigma(i)\}$. The implementation in `src/` [5] comprises four modules:

- `permutations.py` — S_n generation, composition, inversion, cycle notation, subgroup axiom checking
- `stabilizer.py` — stabilizer computation, coset decomposition, Orbit-Stabilizer verification
- `cyclic_group.py` — cyclic subgroup generation, transitivity, isomorphism, non-normality check
- `scheduler_model.py` — unified model and full verification suite

AI assistance. Code, notebooks, and manuscript drafting were assisted by Claude (Anthropic) [4]. All mathematical content was verified independently by the author.

10.2 Verification Strategy

Claim	Strategy
$\|S_n\| = n!$	Enumerate S_n; compare to `math.factorial(n)`
$M(x) \leq S_n$	Filter for $\sigma(x) = x$; verify identity, closure, inverses; check $\|M(x)\| = (n-1)!$ and Lagrange divisibility
$\langle c\rangle \cong \mathbb{Z}_n$	Generate orbit of c; verify $\|\langle c\rangle\| = n$; verify homomorphism $\varphi(j+k \bmod n) = \varphi(j) \circ \varphi(k)$
Deadlock $= e$	Verify e is the unique element satisfying $\sigma(i) = i$ for all $i \in P$

Table 4: Verification strategy per claim.

10.3 Complexity and Feasibility

All algorithms enumerate S_n explicitly, running in $O(n!)$ time and space. This brute-force approach is justified by transparency: every element is constructed and tested individually, leaving no ambiguity about what is verified. The algebraic proofs in Sections 5–8 hold for all n; computation confirms them for small cases.

n	$n!$	Status
6	720	Verified
7	5,040	Feasible, slow
8	40,320	Slow
10	3,628,800	Impractical by enumeration

10.4 Results

n	$\lvert S_n\rvert$	$\lvert M(x)\rvert$	$\lvert\langle c\rangle\rvert$	Deadlock	All pass
2	2	1	2	e	✓
3	6	2	3	e	✓
4	24	6	4	e	✓
5	120	24	5	e	✓
6	720	120	6	e	✓

Table 5: Full verification results.

11 Discussion

11.1 The Subgroup Lattice

The four claims define the following subgroup inclusion structure, shown as a Hasse diagram in Figure 1:

$$\{e\} \leq \langle c\rangle \leq S_n, \qquad \{e\} \leq M(x) \leq S_n.$$

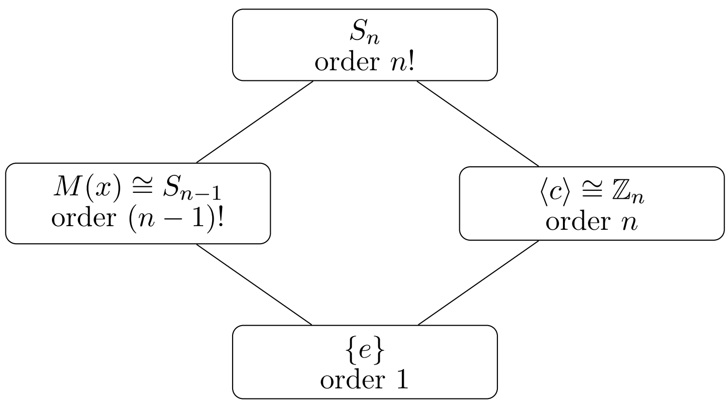

Figure 1: Subgroup lattice (Hasse diagram). An edge from A to B with A below B denotes $A \leq B$. $M(x)$ and $\langle c\rangle$ are incomparable for $n \geq 3$.

11.2 Incomparability of $M(x)$ and $\langle c \rangle$

Proposition 11.1. *For $n \geq 3$, $M(x)$ and $\langle c \rangle$ are incomparable subgroups of S_n, and $M(x) \cap \langle c \rangle = \{e\}$.*

Proof. For $n = 3$, $x = 1$: $M(1) = \{e, (2\ 3)\}$ and $\langle c \rangle = \{e, (1\ 2\ 3), (1\ 3\ 2)\}$. The transposition (2 3) fixes 1, so $(2\ 3) \in M(1)$, but it is not a power of (1 2 3), so $(2\ 3) \notin \langle c \rangle$. The element (1 2 3) satisfies $(1\ 2\ 3)(1) = 2 \neq 1$, so $(1\ 2\ 3) \notin M(1)$. The subgroups are therefore incomparable, and $M(1) \cap \langle c \rangle = \{e\}$.

For $n > 3$: $M(x)$ always contains transpositions $(y\ z)$ with $y, z \neq x$, which have order 2. For odd n, no non-identity power of c has order 2, since the order of c^k is $n/\gcd(n, k)$, which is odd when n is odd. For even n, the unique element of order 2 in $\langle c \rangle$ is $c^{n/2}$, which maps $i \mapsto i + n/2 \pmod n$. In particular, $c^{n/2}(x) = x + n/2 \neq x$ (since $n/2 \not\equiv 0 \pmod n$ for $n \geq 4$), so $c^{n/2} \notin M(x)$. Hence in all cases, the order-2 transpositions in $M(x)$ are not in $\langle c \rangle$, and the incomparability holds. □

OS interpretation. For $n \geq 3$, the mutex-admissible schedules and the round-robin schedules are almost entirely disjoint. Imposing both constraints simultaneously leaves only e — the deadlock state under Assumption 7. This reflects a structural separation between the safety constraint (mutex) and the fairness constraint (round-robin) within the scheduling space.

11.3 Limitations

- **Single epoch.** Dynamic properties such as liveness or starvation freedom require sequences of epochs, which S_n does not model.
- **Deadlock modeling.** Assumption 7 is an operational identification, not a derivation from Coffman conditions. A fuller treatment would require a group-theoretic model of the resource allocation graph.
- **Single critical slot.** Multi-resource mutex requires computing $\bigcap_i \mathrm{Stab}(x_i)$, which is a valid subgroup but is not developed here.

11.4 Possible Extensions

- Multi-resource mutex as $\bigcap_i \mathrm{Stab}(x_i)$
- Readers-writers as a double coset decomposition
- Scheduling histories as paths in the Cayley graph of S_n
- Deadlock detection via a group-theoretic model of the resource allocation graph

12 Conclusion

We have formalized three OS synchronization mechanisms as subgroup-theoretic restrictions on S_n, within the epoch-based model of Section 2, and verified them computationally for $n \leq 6$:

1. The unrestricted scheduling space is S_n with $n!$ elements.

2. Mutual exclusion corresponds to the stabilizer subgroup $M(x) \cong S_{n-1}$ of order $(n-1)!$, with n left cosets partitioning the scheduling space into one admissible region and $n-1$ violation regions.

3. Round-robin scheduling corresponds to the cyclic subgroup $\langle c \rangle \cong \mathbb{Z}_n$ of order n, which is transitive (starvation-free), abelian, and not normal in S_n for $n \geq 3$.

4. Under Assumption 7, deadlock corresponds to the identity element e — the unique permutation under which no process advances.

The subgroups $M(x)$ and $\langle c \rangle$ are incomparable for $n \geq 3$, meeting only at $\{e\}$. This reflects a structural separation between the safety and fairness constraints in the scheduling space. Source code and notebooks are available at [5].

References

[1] A. Silberschatz and P. B. Galvin, *Operating System Concepts*, International Student Version, 8th ed. Wiley India, 2010.

[2] J. A. Gallian, *Contemporary Abstract Algebra*, 8th ed. Cengage Learning, 2017.

[3] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System Deadlocks," *ACM Computing Surveys*, vol. 3, no. 2, pp. 67–78, 1971.

[4] Anthropic, *Claude* (AI assistant, Claude Sonnet 4.6). `https://www.anthropic.com`, 2026. Used for code generation, notebook structure, and manuscript drafting.

[5] O. P. Mohanty, *OS Symmetry Synchronization — Source Code and Notebooks*, GitHub, 2026. `https://github.com/04pranab/os-symmetry-synchronization`