Monotone Operations and Monotone Groups[*]

Costas Busch[†] *Marios Mavronicolas*[‡] *Paul Spirakis*[§]

Abstract

We survey an algebraic approach to proving impossibility results in distributed computing. The approach is emerging around *monotone groups,* a new class of algebraic groups we define here as a tight suit to *monotone* Read&Modify&Write (or RMW) *operations* in a distributed system. The yields of this approach have been the *first* impossibility results for implementations of monotone RMW operations that are based on *switching networks,* a class of concurrent, low-contention data structures in distributed computing.

[*]This work has been partially supported by the Future and Emerging Technologies program of the European Union under EU contracts 001907 (DELIS) and 015964 (AEOLUS), and by research funds from Rensselaer Polytechnic Institute and University of Cyprus.

[†]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180. Email: `buschc@cs.rpi.edu`

[‡]Department of Computer Science, University of Cyprus, Nicosia CY-1678, Cyprus. Email: `mavronic@ucy.ac.cy`

[§]Department of Computer Engineering and Informatics, University of Patras, 265 00 Patras, Greece, & Research and Academic Computer Technology Institute, 261 10 Patras, Greece. Email: `spirakis@cti.gr`

1 Introduction

1.1 Monotone RMW Operations and Monotone Groups

A Read&Modify&Write *shared variable* or *register,* henceforth abbreviated as RMW, is an abstract variable type that allows reading its old value, computing via some specific *operator* a new value as a function of the old value, and writing the new value back to the register, all in a single, *atomic* (indivisible) RMW *operation.*

In this survey, we focus on a specific class of RMW operations whose associated operators correspond to a certain class of algebraic groups introduced by Busch *et al.* [?], which we call *monotone groups.* A monotone group has a *total order* and a *monotone subdomain*; the latter enjoys a significant monotonicity property, which is called *Monotonicity under Composition*: applying the operator on an element from the monotone subdomain results to another element in the monotone subdomain that strictly dominates the initial one with respect to the total order. The Fetch&Add and Fetch&Multiply operations are popular examples from the class. A *monotone* RMW operation [?] is one that is associated with a monotone group.

An abstract concept defined in relation to monotone groups is that of *n-wise independence.* Roughly speaking, n (other than the identity) elements of a monotone group are *n-wise independent* if it is not possible to derive the identity element of the group through some sequence of successive applications of the operator on n of the elements or their inverses. A significant property of monotone groups proved in [?] is that *every* monotone group is *n-wise independent,* in the sense of having n-wise independent elements.

It has been established in [?] that the existence of n-wise independent elements in a monotone group is largely responsible for enforcing *linearizability* [?] for certain suitable executions of a distributed system that implements the corresponding (monotone) RMW operation; recall that an execution is *linearizable* [?] if the values returned to operations in it respect the real-time ordering of the operations. Indeed, the main conclusion of Busch *et al.* [?] is that the requirement to guarantee the *inherent* linearizability for certain particular executions incurs a high cost in efficiency for a certain class of highly concurrent, low-contention implementations of (monotone) RMW that are based on *switching networks* [?].

1.2 The Monotone Linearizability Lemma

The *Monotone Linearizability Lemma* [?, Proposition 5.1] (repeated here as Proposition ??) establishes inherent ordering constraints of linearizability for a certain class of executions of *any* distributed system that implements a monotone RMW operation. Interestingly, in these executions, the arguments of the RMW operations performed by the n participating concurrent processes enjoy together the group-theoretic property of n-wise independence over the associated monotone group.

In order to gain some intuition for the *Monotone Linearizability Lemma* and its proof, we offer a substantially simpler proof for a corresponding *Monotone Sequential Consistency Lemma* (Proposition **??**) that we prove here. In a corresponding way, the *Monotone Sequential Consistency Lemma* establishes inherent ordering constrains of *sequential consistency* [**?**] for a certain class of executions of *any* distributed system that implements a monotone RMW operation. (Recall that an execution is *sequentially consistent* [**?**] if the values returned to operations at the *same* process respect the real-time ordering of the operations.)

1.3 Switching Networks

The Monotone Linearizability Lemma has been applied to implementations of monotone RMW operations based on switching networks. The application has yielded the *first* lower bounds on size for any highly concurrent, low-contention switching network that implements a monotone RMW operation.

More specifically, Busch *et al.* [**?**] have obtained the following results for any switching network other than the trivial single-switch one:

- If the switching network is made up of switches with a *finite* number of states and it is low-contention, then it must contain an *infinite* number of switches, even if concurrency is restricted to remain *bounded* (Theorem **??**).

- If the switching network is made up of switches with an *infinite* number of states and it is low-contention, then it must still contain an *infinite* number of switches if concurrency is now allowed to grow unbounded (Theorem **??**).

2 Monotone Groups

In this section, we review monotone groups, closely following [**?**, Section 2], where all definitions and results come from. Section **??** reviews some very basic definitions from Group Theory.[*] Some composite operators are introduced in Section **??**. Section **??** provides the basic definitions for monotone groups. Pairwise independence is introduced in Section **??**, while Section **??** establishes that all monotone groups are pairwise independent. Similarly, n-wise independence is introduced in Section **??**, while Section **??** establishes that all monotone groups are n-wise independent.

Throughout this section (and in the rest of the paper), denote Z, $\mathbb{N}$ and Q the sets of integers, natural numbers (including zero), and rational numbers (excluding zero), respectively.

[*]The interested reader may consult [**?**] for a general background in Group Theory.

We will use $+$ and $\cdot$ to denote the common (binary) operators of addition and multiplication, respectively, on these sets. Denote $\leq$ the *less-than-or-equal* relation (total order) on these sets.

2.1 Groups and Abelian Groups

A (binary) *operator* (also called *composition law*) on a set $\mathbb{\Gamma}$ is a mapping $\oplus : \mathbb{\Gamma} \times \mathbb{\Gamma} \rightarrow \mathbb{\Gamma}$. A *group* $\langle \mathbb{\Gamma}, \oplus \rangle$ is a set $\mathbb{\Gamma}$, sometimes called the *ground set,* together with an operator $\oplus$ such that the following properties hold:

1. *Closure:* For all pairs of elements $a, b \in \mathbb{\Gamma}$, $a \oplus b \in \mathbb{\Gamma}$.

2. *Associativity:* For all triples of elements $a, b, c \in \mathbb{\Gamma}$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.

3. *Identity Element:* There is an element $a \in \mathbb{\Gamma}$, called the *identity element* of $\mathbb{\Gamma}$, such that for each element $a \in \mathbb{\Gamma}$, $a \oplus e = e \oplus a = a$.

4. *Inverse Element:* for each element $a \in \mathbb{\Gamma}$, there is an element $a^{-1} \in \mathbb{\Gamma}$, called the *inverse* of a, such that $a \oplus a^{-1} = a^{-1} \oplus a = e$.

An *Abelian group* is a group $\langle \mathbb{\Gamma}, \oplus \rangle$ which satisfies the following additional property:

5. *Commutativity:* For all pairs of elements $a, b \in \mathbb{\Gamma}$, $a \oplus b = b \oplus a$.

Note that $e^{-1} = e$. Note also that for any sequence of elements $a, b, \ldots \in \mathbb{\Gamma}$, the *Associativity* property implies that $(a \oplus b \oplus \ldots \oplus w)^{-1} = w^{-1} \oplus \ldots \oplus b^{-1} \oplus a^{-1}$. Finally, the following elementary property of groups will be used in some of our later proofs.

Property 2.1 (Cancellation Law) *Consider any group $\langle \mathbb{\Gamma}, \oplus \rangle$. Then, for any triple of elements $a, b, c \in \mathbb{\Gamma}$, $a \oplus b = a \oplus c$ (resp., $b \oplus a = c \oplus a$) implies $b = c$.*

2.2 Composite Operators

We proceed to define two composite operators by applying the (binary) operator $\oplus$ a number of times.

2.2.1 The Power Operator

For any integer k, define the unary operator $\bigoplus_k : \mathbb{\Gamma} \rightarrow \mathbb{\Gamma}$ as follows:

$$\bigoplus_k a \quad = \quad \begin{cases} \underbrace{a \oplus a \oplus \ldots \oplus a}_{k \text{ times}}, & \text{if } k > 0 \\ e\,, & \text{if } k = 0 \\ \underbrace{a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1}}_{-k \text{ times}}, & \text{if } k < 0 \end{cases}$$

Call $\bigoplus_k$ the *power operator*. For any element $a \in \Gamma$, use the power operator $\bigoplus_k$ defined for all integers k to define the set $\Gamma_a = \{\bigoplus_k a \mid k \in Z\}$. Thus, $\langle \Gamma_a, \oplus \rangle$ is a *cyclic* group with *generator* a.

By the definition for the power operator, it follows that for any element $a \in \Gamma$ and integer k, $\bigoplus_k a = \bigoplus_{-k} a^{-1}$. We continue to prove two elementary properties of the power operator that will be used in some of our later proofs.

Property 2.2 (Superposition of Powers) *For any Abelian group $\langle \Gamma, \oplus \rangle$, fix any element $a \in \Gamma$. Then, for any sequence of integers $k_1, k_2, \ldots, k_n$,*

$$\left(\bigoplus_{k_1} a\right) \oplus \left(\bigoplus_{k_2} a\right) \oplus \ldots \oplus \left(\bigoplus_{k_n} a\right) = \bigoplus_{\sum_{i=1}^{n} k_i} a\,.$$

Proof: By definition of the power operator, each factor $\bigoplus_{k_i} a$, $1 \leq i \leq n$, contributes:

- either the element a k_i times if $k_i > 0$ (call these *positive* contributions),
- or the element a^{-1} $-k_i$ times if $k_i < 0$ (call these *negative* contributions),
- or the identity element e if $k_i = 0$ (call these *zero* contributions),

to the composite expression $(\bigoplus_{k_1} a) \oplus (\bigoplus_{k_2} a) \oplus \ldots \oplus (\bigoplus_{k_n} a)$. By the *Commutativity* property, positive, negative and zero contributions can be separated from each other in the composite expression. By definition of the power operator, all zero contributions result to e; by the *Associativity* property, each pair of a positive and a negative contribution cancels out. It follows that the composite expression simplifies to $\bigoplus_{\sum_{i=1}^{n} k_i} a$, as needed. ■

We finally prove:

Property 2.3 (Composition of Powers) *For any group $\langle \Gamma, \oplus \rangle$, fix any element $a \in \Gamma$. Then, for any integer k and natural number l, $\bigoplus_k (\bigoplus_l a) = \bigoplus_{k \cdot l} a$.*

Proof: Consider first the case $l = 0$.

- On one hand, $\bigoplus_l a = \bigoplus_0 a = e$ (by definition of the power operator); so,

$$\begin{aligned}
\bigoplus_k \left(\bigoplus_l a\right) &= \bigoplus_k e \\
&= \begin{cases} \underbrace{e \oplus e \oplus \ldots \oplus e}_{k \text{ times}}, & \text{if } k > 0 \\ e\,, & \text{if } k = 0 \\ \underbrace{e^{-1} \oplus e^{-1} \oplus \ldots \oplus e^{-1}}_{-k \text{ times}}, & \text{if } k < 0 \end{cases} \\
&= e\,;
\end{aligned}$$

- On the other hand, $\bigoplus_{k\cdot l} a = \bigoplus_0 a = e$ (by definition of the power operator).

It follows that $\bigoplus_k \left(\bigoplus_l a\right) = \bigoplus_{k\cdot l} a$ for $l = 0$. So assume $l > 0$. Clearly,

$$
\begin{aligned}
& \bigoplus_k \left(\bigoplus_l a\right) \\
= & \begin{cases} \underbrace{\left(\bigoplus_l a\right) \oplus \left(\bigoplus_l a\right) \oplus \ldots \oplus \left(\bigoplus_l a\right)}_{k \text{ times}}, & \text{if } k > 0 \\ e, & \text{if } k = 0 \\ \underbrace{\left(\bigoplus_l a\right)^{-1} \oplus \left(\bigoplus_l a\right)^{-1} \oplus \ldots \oplus \left(\bigoplus_l a\right)^{-1}}_{-k \text{ times}}, & \text{if } k < 0 \end{cases} \\
= & \begin{cases} \underbrace{\underbrace{(a \oplus a \oplus \ldots \oplus a)}_{l \text{ times}} \oplus \underbrace{(a \oplus a \oplus \ldots \oplus a)}_{l \text{ times}} \oplus \ldots \oplus \underbrace{(a \oplus a \oplus \ldots \oplus a)}_{l \text{ times}}}_{k \text{ times}}, & \text{if } k > 0 \\ e, & \text{if } k = 0 \\ \underbrace{\underbrace{(a \oplus a \oplus \ldots \oplus a)^{-1}}_{l \text{ times}} \oplus \underbrace{(a \oplus a \oplus \ldots \oplus a)^{-1}}_{l \text{ times}} \oplus \ldots \oplus \underbrace{(a \oplus a \oplus \ldots \oplus a)^{-1}}_{l \text{ times}}}_{-k \text{ times}}, & \text{if } k < 0 \end{cases} \\
= & \begin{cases} \underbrace{a \oplus a \oplus \ldots \oplus a}_{k \cdot l \text{ times}}, & \text{if } k > 0 \\ e, & \text{if } k = 0 \\ \underbrace{\underbrace{(a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1})}_{l \text{ times}} \oplus \underbrace{(a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1})}_{l \text{ times}} \oplus \ldots \oplus \underbrace{(a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1})}_{l \text{ times}}}_{-k \text{ times}}, & \text{if } k < 0 \end{cases} \\
= & \begin{cases} \underbrace{a \oplus a \oplus \ldots \oplus a}_{k \cdot l \text{ times}}, & \text{if } k > 0 \\ e, & \text{if } k = 0 \\ \underbrace{a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1}}_{-k \cdot l \text{ times}}, & \text{if } k < 0 \end{cases} \\
= & \bigoplus_{k \cdot l} a\,,
\end{aligned}
$$

as needed. ■

2.2.2 The Summation Operator

For any integer n, the operator $\biguplus_n : \mathbb{\Gamma}^n \to \mathbb{\Gamma}$ is n-ary.

- For $n = 0$, it assumes the constant value $\biguplus_0 = e$.

- For $n = 1$, $\biguplus_1\{a\} = a$ for all elements $a \in \mathbb{\Gamma}$. For $n = -1$, $\biguplus_{-1}\{a\} = a^{-1}$.
- For $|n| \geq 2$. $\biguplus_n$ takes as input an ordered multiset of elements $\{a_1, a_2, \ldots, a_{|n|}\} \in \mathbb{\Gamma}$, and it yields the result

$$\biguplus_n \{a_1, a_2, \ldots, a_n\} \quad = \quad \begin{cases} a_1 \oplus a_2 \oplus \ldots \oplus a_{|n|}\,, & \text{if } n \geq 2 \\ a_1^{-1} \oplus a_2^{-1} \oplus \ldots \oplus a_{|n|}^{-1}\,, & \text{if } n \leq -2 \end{cases}$$

  denoted also as $\biguplus_{i=1}^{n} a_i$. Note that, by associativity, the result of applying the operator is well defined.

Call $\biguplus$ the *summation operator*. Our definitions for the power and summation operators immediately imply that for any element $a \in \mathbb{\Gamma}$ and for any integer $n \neq 0$,

$$\bigoplus_n a \quad = \quad \begin{cases} \biguplus_n \left\{ \underbrace{a, a, \ldots, a}_{n \text{ times}} \right\}, & \text{if } n > 0 \\ \biguplus_n \left\{ \underbrace{a, a, \ldots, a}_{-n \text{ times}} \right\}, & \text{if } n < 0 \end{cases}$$

So, roughly speaking, the power operator is some special case of the summation operator where all inputs are identical.

The result $\biguplus_n \{a_1, a_2, \ldots, a_n\}$ of the summation operator will sometimes be called a *composite expression.*

2.3 Monotone Groups

Assume now that the set $\mathbb{\Gamma}$ is totally ordered;[†] thus, a *total order* $\preceq$ is defined on $\mathbb{\Gamma}$. For any pair of elements $a, b \in \mathbb{\Gamma}$, write $a \prec b$ (and, equivalently, $b \succ a$) if $a \preceq b$ and $a \neq b$.

A *monotone subdomain* of $\mathbb{\Gamma}$ is a subset $\mathbb{M} \subseteq \mathbb{\Gamma}$ that satisfies the following three properties:

1. *Closure:* For any two elements $a, b \in \mathbb{M}$, $a \oplus b \in \mathbb{M}$.
2. *Identity Lower Bound:* For any element $a \in \mathbb{M}$, $e \prec a$.
3. *Monotonicity under Composition:* For any pair of elements $a, b \in \mathbb{M}$, both $a \prec a \oplus b$ and $b \prec a \oplus b$.

[†]The idea of augmenting the ground set of group with some order (total or partial) has also been followed in defining and studying partially ordered algebraic structures, research on which has burgeoned in the last 50 years. However, the resulting *partially ordered groups* [?] are completely different than the *monotone groups* introduced and studied in this work.

Notice that the *Identity Lower Bound* property used in the definition of the monotone subdomain $\mathbb{M}$ implies that $e \notin \mathbb{M}$, so that $\mathbb{M} \subset \mathbb{\Gamma}$. Notice also that the *Monotonicity under Composition* property used in the definition of the monotone subdomain $\mathbb{M}$ implies that $\mathbb{M}$ is necessarily infinite. We are now ready to define monotone groups.

Definition 2.1 (Monotone Group) *A* monotone group *is a quadruple* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$, *where* $\langle \mathbb{\Gamma}, \oplus \rangle$ *is an Abelian group,* $\preceq$ *is a total order on* $\mathbb{\Gamma}$, *and* $\mathbb{M} \subseteq \mathbb{\Gamma}$ *is a monotone subdomain of* $\mathbb{\Gamma}$.

We proceed with some examples of monotone groups that will be used in our later analysis.

Example 2.1 *The quadruple* $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq \rangle$.

Clearly, the quadruple $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq \rangle$ is a monotone group, called *Integers with Addition.* It is associated with the monotone Fetch&Add operation.

- ¿From the definition of the power operator $\bigoplus_k$, for any integer k, we have that for any integer $a \in \mathrm{Z}$, $\bigoplus_k a = k \cdot a$.
- ¿From the definition of the summation operator $\biguplus_{k_1}^{k_2}$, for any pair of integers k_1 and k_2, we have that for any sequence of $k_2 - k_1 + 1$ integers $a_{k_1}, a_{k_1+1}, \ldots, a_{k_2} \in \mathrm{Z}$,

$$\biguplus_{i=k_1}^{k_2} a_i \quad = \quad \sum_{i=k_1}^{k_2} a_i \,.$$

$\square$

Example 2.2 *The quadruple* $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle$.

Clearly, the quadruple $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle$ is also a monotone group, called *Rationals with Multiplication.* It is associated with the monotone Fetch&Multiply operation.

- ¿From the definition of the power operator $\bigoplus_k$, for any integer k, we have that for any rational number $a \in \mathrm{Q}$, $\bigoplus_k a = a^k$.
- ¿From the definition of the summation operator $\biguplus$, for any set of n integers $k_1, k_2, \ldots, k_n$, we have that for any set of n rational numbers $a_{k_1}, a_{k_1+1}, \ldots, a_{k_2} \in \mathrm{Q}$,

$$\biguplus_{i=k_1}^{k_2} a_i \quad = \quad \prod_{i=k_1}^{k_2} a_i \,.$$

□

We finally prove an elementary, non-idempotency property of monotone groups that will be used in some of our later proofs.

Property 2.4 (No Idempotent Power) *For any arbitrary monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$, fix any element $a \in \mathbb{M}$. Then, for any integer k, $\bigoplus_k a = e$ implies $k = 0$.*

Proof: Assume, by way of contradiction, that $k \neq 0$. Consider first the case where $k > 0$. Then,

$$\begin{array}{lll} & \bigoplus_k a & \\ = & \underbrace{a \oplus a \oplus \ldots \oplus a}_{k \text{ times}} & \text{(by definition of the power operator)} \\ \succ & e & \text{(by \textit{Monotonicity under Composition})}\ , \end{array}$$

a contradiction.

Consider now the case where $k < 0$. By definition of the power operator and associativity, it follows that

$$\begin{array}{rcl} \underbrace{a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1}}_{-k \text{ times}} & = & e \\ & = & \left(\underbrace{a^{-1} \oplus a^{-1} \oplus \ldots \oplus a^{-1}}_{-k \text{ times}} \right) \oplus \left(\underbrace{a \oplus a \oplus \ldots \oplus a}_{-k \text{ times}} \right) . \end{array}$$

By Property **??**, it follows that

$$e = \underbrace{a \oplus a \oplus \ldots \oplus a}_{-k \text{ times}} .$$

Since $-k > 0$, we are reduced to the first case, and the proof is now complete. ■

We remark that the proof of Property **??** relied on using the *Monotonicity under Composition* property that holds specifically for monotone groups. So, it is no coincidence that Property **??** does *not* necessarily hold for a *general* group.

2.4 Pairwise Independence

Consider any two distinct elements $a_1, a_2 \in \mathbb{\Gamma}$ with $a_1, a_2 \neq e$. Say that a_1 and a_2 are *pairwise independent over* $\langle \mathbb{\Gamma}, \oplus \rangle$ if for any integer k, both $a_1 \neq \bigoplus_k a_2$ and $a_2 \neq \bigoplus_k a_1$. Thus, neither a_1 nor a_2 may result by repetitive application of the operator on the other or on the inverse of the other. We are now ready to define a pairwise independent monotone group.

Definition 2.2 (Pairwise Independent Monotone Group) *Say that the monotone group* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ *is* pairwise independent *if there are two distinct elements* $a_1, a_2 \in \mathbb{M}$*, with* $a_1, a_2 \neq e$*, that are pairwise independent over* $\langle \mathbb{\Gamma}, \oplus \rangle$.

We continue with two examples of pairwise independent monotone groups.

Example 2.3 *Pairwise independence of the monotone group* $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq \rangle$.

¿From the definition of pairwise independence, two integers $a_1, a_2 \in \mathrm{Z}$, with $a_1, a_2 \neq 0$, are pairwise independent in $\langle \mathrm{Z}+ \rangle$ if for any integer k, both $a_1 \neq k \cdot a_2$ and $a_2 \neq k \cdot a_1$. For any integer $a \geq 2$, consider the consecutive natural numbers a and $a+1$; so, $a, a+1 \in \mathbb{N} \setminus \{0\}$. Clearly, for any integer k, both $a \neq k \cdot (a+1)$ and $a+1 \neq k \cdot a$. It follows that a and $a+1$ are pairwise independent over $\langle \mathrm{Z}, + \rangle$. Hence, the monotone group $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq \rangle$ is pairwise independent. □

Example 2.4 *Pairwise independence of the monotone group* $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle$.

¿From the definition of pairwise independence, two rational numbers $a_1, a_2 \in \mathrm{Q}$, with $a_1, a_2 \neq 1$, are pairwise independent in $\langle \mathrm{Q}, \cdot \rangle$ if for any integer k, both $a_1 \neq a_2^k$ and $a_2 \neq a_1^k$. Consider any pair of distinct prime numbers p_1 and p_2 greater than 1; so $p_1, p_2 \in \mathbb{N} \setminus \{0, 1\}$. Clearly, for any integer k, both $p_1 \neq p_2^k$ and $p_2 \neq p_1^k$. It follows that p_1 and p_2 are pairwise independent over $\langle \mathrm{Q}, \cdot \rangle$. Hence, the monotone group $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle$ is pairwise independent. □

2.5 Pairwise Independence of All Monotone Groups

We prove that *every* monotone group is pairwise independent.

Lemma 2.5 (Every Monotone Group is Pairwise Independent) *Every monotone group* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ *is pairwise independent.*

Proof: Since the monotone group $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq \rangle$ is pairwise independent (see Example **??**), there exist *distinct* natural numbers l_1 and l_2 in $\mathbb{N} \setminus \{0\}$ that are pairwise independent over $\langle \mathrm{Z}, + \rangle$. Fix now any arbitrary element $a \in \mathbb{M}$, and consider the elements $\bigoplus_{l_1} a$ and $\bigoplus_{l_2} a$. Clearly, by monotonicity under composition, these two elements are distinct. We will prove that $\bigoplus_{l_1} a$ and $\bigoplus_{l_2} a$ are pairwise independent over $\langle \mathbb{\Gamma}, \oplus \rangle$.

Assume, by way of contradiction, that the elements $\bigoplus_{l_1} a$ and $\bigoplus_{l_2} a$ are *not* pairwise independent over $\langle \mathbb{\Gamma}, \oplus \rangle$. So, there exists some integer k such that either $\bigoplus_{l_1} a = \bigoplus_k \left(\bigoplus_{l_2} a\right)$ or $\bigoplus_{l_2} a = \bigoplus_k \left(\bigoplus_{l_1} a\right)$. Assume, without loss of generality, that $\bigoplus_{l_1} a = \bigoplus_k \left(\bigoplus_{l_2} a\right)$. By Property **??**, this implies that $\bigoplus_{l_1} a = \bigoplus_{k \cdot l_2} a$. It follows that $\left(\bigoplus_{l_1} a\right) \oplus \left(\bigoplus_{-k \cdot l_2} a\right) = \left(\bigoplus_{k \cdot l_2} a\right) \oplus$

$(\bigoplus_{-k\cdot l_2} a)$. By Property **??**, this implies that $\bigoplus_{l_1-k\cdot l_2} a = \bigoplus_{k\cdot l_2 - k\cdot l_2} a$, or $\bigoplus_{l_1-k\cdot l_2} a = e$. By Property **??**, it follows that $l_1 - k\cdot l_2 = 0$ or $l_1 = k\cdot l_2$. Thus, the natural numbers l_1 and l_2 are *not* pairwise independent over $\langle \mathrm{Z}, +\rangle$. A contradiction. ■

2.6 n-Wise Independence

Fix any integer $n \geq 2$. Consider any n distinct elements $a_1, a_2, \ldots, a_n \in \mathbb{\Gamma}$ with $a_1, a_2, \ldots, a_n \neq e$. Say that $a_1, a_2, \ldots, a_n$ are *n-wise independent over* $\langle \mathbb{\Gamma}, \oplus\rangle$ if for any sequence of n integers $k_1, k_2, \ldots, k_n$, where $-1 \leq k_i \leq 2$ for $1 \leq i \leq n$, that are *not all simultaneously* zero, $\biguplus_{i=1}^{n} \bigoplus_{k_i} a_i \neq e$. We are now ready to define an n-wise independent monotone group.

Definition 2.3 (n-Wise Independent Monotone Group) *Say that the monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq\rangle$ is* n-wise independent *if there are n distinct elements $a_1, a_2, \ldots, a_n \in \mathbb{M}$, with $a_1, a_2, \ldots, a_n \neq e$, that are n-wise independent over $\langle \mathbb{\Gamma}, \oplus\rangle$.*

Note that n-wise independence is not just a trivial generalization (from 2 to n) of pairwise independence, since it imposes constraints on the integers k_i, $1 \leq i \leq n$ (namely, that $-1 \leq k_i \leq 2$). Notice that, in particular, 2-wise independence and pairwise independence are not identical concepts.

¿From the definition of n-wise independence, n integers $a_1, a_2, \ldots, a_n \in \mathrm{Z}$, where $n \geq 2$, are n-wise independent over $\langle \mathrm{Z}, +\rangle$ if for any sequence of n integers $k_1, k_2, \ldots, k_n \in \{-1, 0, 1, 2\}$, which are not all simultaneously zero, $\sum_{i=1}^{n} k_i \cdot a_i \neq 0$.

We prove that for any integer $n \geq 2$, the monotone group $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq\rangle$ is n-wise independent.

Lemma 2.6 *For any integer $n \geq 2$, the monotone group $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq\rangle$ is n-wise independent.*

Proof: Fix any integer $\ell \geq 0$. Consider the n natural numbers $2^{\ell}, 2^{\ell+2}, \ldots, 2^{\ell+2(n-1)} \in \mathbb{N}\setminus\{0\}$, which are powers of two; we will prove that these n natural numbers are n-wise independent over $\langle \mathrm{Z}, +\rangle$. The proof is by induction on n.

For the basis case where $n = 2$, consider the natural numbers 2^{ℓ} and $2^{\ell+2}$. Fix any pair of integers $k_1, k_2 \in \{-1, 0, 1, 2\}$ that are not both simultaneously zero. Clearly, $k_1 2^{\ell} + k_2 2^{\ell+2} = 2^{\ell}(k_1 + 4k_2)$, which can be zero only if $k_1 = k_2 = 0$. So, the natural numbers $2^{\ell}, 2^{\ell+2} \in \mathbb{N} \setminus \{0\}$ are 2-wise independent over $\langle \mathrm{Z}, +\rangle$. Hence, the monotone group $\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq\rangle$ is 2-wise independent. This completes the proof of the basis case.

Assume inductively that the $n-1$ natural numbers $2^{\ell}, 2^{\ell+2}, \ldots, 2^{\ell+2((n-1)-1)} = 2^{\ell+2(n-2)} \in \mathbb{N} \setminus \{0\}$ are $(n-1)$-wise independent over $\langle \mathrm{Z}, +\rangle$.

For the induction step, we will show that the n natural numbers $2^{\ell}, 2^{\ell+2}, \ldots, 2^{\ell+2(n-1)}$ are n-wise independent in $\langle \mathrm{Z}, + \rangle$. Assume, by way of contradiction, that they are not. Thus, there exist n integers $k_1, k_2, \ldots, k_n \in \{-1, 0, 1, 2\}$ which are not all simultaneously zero, such that $\sum_{i=1}^{n} k_i 2^{\ell+2(i-1)} = 0$. We proceed by case analysis on the value of $k_n \in \{-1, 0, 1, 2\}$.

- Assume first that $k_n = -1$. Then, $\sum_{i=1}^{n-1} k_i 2^{\ell+2(i-1)} - 2^{\ell+2(n-1)} = 0$, or $\sum_{i=1}^{n-1} k_i 2^{\ell+2(i-1)} = 2^{\ell+2(n-1)}$, or $\sum_{i=1}^{n-1} k_i 2^{2(i-1)} = 2^{2(n-1)}$. However, since $k_i \leq 2$ for all indices i, $1 \leq i \leq n-1$,

$$\begin{aligned} \sum_{i=1}^{n-1} k_i 2^{2(i-1)} &\leq 2 \sum_{i=1}^{n-1} 2^{2(i-1)} \\ &< 2 \cdot \sum_{i=0}^{2n-4} 2^{i} \\ &= 2\left(2^{2n-3} - 1\right) < 2^{2n-2} = 2^{2(n-1)}, \end{aligned}$$

  a contradiction.

- Assume now that $k_n = 0$. Then, $\sum_{i=1}^{n-1} k_i 2^{\ell+2(i-1)} = 0$. Since the integers $k_1, k_2, \ldots, k_n$ are not all simultaneously zero while $k_n = 0$, it follows that the integers $k_1, k_2, \ldots, k_{n-1}$ are not all simultaneously zero. This implies that the $n-1$ natural numbers $2^{\ell}, 2^{\ell+2}, \ldots, 2^{\ell+2(n-2)}$ are $(n-1)$-wise independent over $\langle \mathrm{Z}, + \rangle$, which contradicts the induction hypothesis.

- Assume finally that $k_n \in \{1, 2\}$. Then, $\sum_{i=1}^{n-1} k_i 2^{\ell+2(i-1)} + k_n \cdot 2^{\ell+2(n-1)} = 0$, or, equivalently, $-\sum_{i=1}^{n-1} k_i 2^{\ell+2(i-1)} = k_n \cdot 2^{\ell+2(n-1)}$, or $-\sum_{i=1}^{n-1} k_i 2^{2(i-1)} = k_n \cdot 2^{2(n-1)}$. However, since $k_i \geq -1$ for all indices i, $1 \leq i \leq n-1$,

$$\begin{aligned} -\sum_{i=1}^{n-1} k_i 2^{2(i-1)} &\leq \sum_{i=1}^{n-1} 2^{2(i-1)} \\ &< \sum_{i=0}^{2n-4} 2^{i} \\ &= 2^{2n-3} - 1 < 2^{2n-2} = 2^{2(n-1)} \leq k_n \cdot 2^{2(n-1)}, \end{aligned}$$

  a contradiction.

Since we obtained a contradiction in all possible cases, the proof is now complete. ■

We finally prove:

Lemma 2.7 *For any integer $n \geq 2$, the monotone group $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle$ is n-wise independent.*

Proof: ¿From the definition of n-wise independence, n rational numbers $a_1, a_2, \ldots, a_n \in \mathrm{Q}$ are n-wise independent over $\langle \mathrm{Q}, \cdot \rangle$ if for any sequence of n integers $k_1, k_2, \ldots, k_n \in \{-1, 0, 1, 2\}$, which are not all simultaneously zero, $\prod_{i=1}^{n} a_i^{k_i} \neq 1$.

Consider any n distinct prime numbers $p_1, p_2, \ldots, p_n \in \mathbb{N} \setminus \{0, 1\}$. Then, $\prod_{i=1}^{n} p_i^{k_i}$ is a rational number whose numerator and denominator have no common factors; so $\prod_{i=1}^{n} p_i^{k_i} \neq 1$. Thus, the n prime numbers $p_1, p_2, \ldots, p_n$ are n-wise independent over Q. Hence, the monotone group $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle$ is n-wise independent, as needed. ■

2.7 n-Wise Independence of All Monotone Groups

We finally prove that *every* monotone group is n-wise independent.

Lemma 2.8 (Every Monotone Group is n-Wise Independent) *For any integer $n \geq 2$, the monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ is n-wise independent.*

Proof: Since the monotone group $\langle \mathrm{Q}, \mathbb{N} \setminus \{0\}, +, \leq \rangle$ is n-wise independent (Lemma **??**), there exist n distinct natural numbers $l_1, l_2, \ldots, l_n \in \mathbb{N} \setminus \{0\}$ that are n-wise independent over $\langle \mathrm{Z}, + \rangle$. Fix any element $a \in \mathbb{M}$. and consider the n elements $\bigoplus_{l_1} a, \bigoplus_{l_2} a, \ldots, \bigoplus_{l_n} a$ of $\mathbb{M}$. Clearly, by the *Monotonicity under Composition* property of the monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$, these n elements are distinct. We will prove that they are also n-wise independent over $\langle \mathbb{\Gamma}, \oplus \rangle$.

Assume, by way of contradiction, that the elements $\bigoplus_{l_1} a, \bigoplus_{l_2} a, \ldots, \bigoplus_{l_n} a$ are *not* n-wise independent over $\langle \mathbb{\Gamma}, \oplus \rangle$. Thus, there exist n integers $k_1, k_2, \ldots, k_n \in \{-1, 0, 1, 2\}$, which are not all simultaneously zero, such that

$$\biguplus_{i=1}^{n} \left(\bigoplus_{k_i} \left(\bigoplus_{l_i} a \right) \right) \quad = \quad e\,.$$

By Property **??**, it follows that

$$\biguplus_{i=1}^{n} \left(\bigoplus_{k_i \cdot l_i} a \right) \quad = \quad e\,,$$

which, by the definition of the summation operator, may be written as

$$\left(\bigoplus_{k_1 \cdot l_1} a \right) \oplus \left(\bigoplus_{k_2 \cdot l_2} a \right) \oplus \ldots \oplus \left(\bigoplus_{k_n \cdot l_n} \right) \quad = \quad e\,.$$

By Property **??**, it follows that

$$\bigoplus_{\sum_{i=1}^{n} k_i \cdot l_i} a \quad = \quad e\,.$$

Property **??**, now implies that $\sum_{i=1}^{n} k_i \cdot l_i = 0$. Since the integers k_i, $1 \leq i \leq n$, are from the set $\{-1, 0, 1, 2\}$, and they are not all simultaneously zero, this implies that the n natural numbers $l_1, l_2, \ldots, l_n$ are *not* n-wise independent over $\langle \mathrm{Z}, + \rangle$. A contradiction. ■

We remark that the proof of Lemma **??** employs the n-wise independence of the monotone group $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, \}, +, \leq \rangle$ (which was established in Lemma **??**) in order to conclude the n-wise independence of the arbitrary monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \leq \rangle$. So, this proof by reduction indicates some kind of completeness of the monotone group $\langle \mathrm{Q}, \mathbb{N} \setminus \{0, \}, +, \leq \rangle$ for the class of all n-wise independent monotone groups.

3 Distributed System

Our model of a distributed system is patterned after the one in [**?**, Section 2], adjusted to incorporate the issue of implementing a monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$.

We consider a distributed system $\mathbf{P}$ consisting of a collection of sequential threads of control, called *processes*. Processes are sequential, and each process applies a sequence of operations to a distributed data structure, called the *object,* alternately issuing an invocation and then receiving the associated response. Each *invocation* at process p_i has the form $\texttt{Invoke}_i(a)$ for some value $a \in \mathbb{M}$; each *response* at process p_i has the form $\texttt{Response}_i(b)$ for some value $b \in \mathbb{M} \cup \{e\}$.

Formally, an *execution* of system $\mathbf{P}$ is a (possibly infinite) sequence α of *invocation* and *response* events. We assume that for each invocation at process p_i in execution α, there is a later response in α that matches it and no invocation at p_i that precedes the matching response in α. Prefixes and suffixes of an execution are defined in the natural way. Say that an execution γ *extends* a prefix β of execution α if β is a prefix of γ as well.

An *operation* at process p_i in execution α is a matching pair $op_i = [\texttt{Invoke}_i(a), \texttt{Response}_i(b)]$ of an invocation and response at p_i; we will sometimes say that op_i is of *type* a. For such an operation, we will write $a = \mathsf{In}(op_i)$ and $b = \mathsf{Out}(op_i)$; thus, op_i has *input* and *output* a and b, respectively.

An execution α induces a partial order $\xrightarrow{\alpha}$ on the set of operations in α as follows.

For any pair of operations

$$op_{i_1} \quad = \quad [\texttt{Invoke}_{i_1}(a_1), \texttt{Response}_{i_1}(b_1)]$$

and

$$op_{i_2} \quad = \quad [\texttt{Invoke}_{i_2}(a_2), \texttt{Response}_{i_2}(b_2)]$$

at processes p_{i_1} and p_{i_2}, respectively, say that op_{i_1} *precedes* op_{i_2} *in execution* α, denoted $op_{i_1} \xrightarrow{\alpha} op_{i_2}$, if the response $\texttt{Response}_{i_1}(b_1)$ precedes the invocation $\texttt{Invoke}_{i_2}(a_2)$.

In particular, execution α induces, for each process p_i a total order $\xrightarrow{\alpha}_i$ on the set of operations at p_i in α as follows.

For any two operations $op_i^{(1)}$ and $op_i^{(2)}$, $op_i^{(1)} \xrightarrow{\alpha}_i op_i^{(2)}$ if and only if $op_i^{(1)} \xrightarrow{\alpha} op_i^{(2)}$.

If, in execution α, operation op_{i_1} does not precede operation op_{i_2}, then we write $op_{i_1} \overset{\alpha}{\not\longrightarrow} op_{i_2}$. If simultaneously $op_{i_1} \overset{\alpha}{\not\longrightarrow} op_{i_2}$ and $op_{i_2} \overset{\alpha}{\not\longrightarrow} op_{i_1}$, then we say that op_{i_1} and op_{i_2} are *parallel* in execution α, denotedk as $op_{i_1} \parallel_\alpha op_{i_2}$.

For any execution α of system $\mathbf{P}$, a *serialization* $S(\alpha)$ of execution α is a sequence whose elements are the operations of α, and each operation of α appears exactly once in $S(\alpha)$. Thus, a serialization $S(\alpha)$ is a total order $\xrightarrow{S(\alpha)}$ on the set of operations in α. Notice that there may be, in general, many possible serializations of the execution α.

Say that a serialization $S(\alpha)$ is *valid for the monotone group* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ if the following two conditions hold:

1. *Valid Start:* If $op_i = [\texttt{Invoke}_i(a), \texttt{Response}_i(b)]$ is the first operation in $S(\alpha)$, then $b = e$.
2. *Valid Composition:* For any pair of operations $op_{i_1}^{(1)} = [\texttt{Invoke}_{i_1}(a_1), \texttt{Response}_{i_1}(b_1)]$ and $op_{i_2}^{(2)} = [\texttt{Invoke}_{i_2}(a_2), \texttt{Response}_{i_2}(b_2)]$ that are consecutive in $S(\alpha)$, $b_2 = b_1 \oplus a_1$.

Sometimes we shall simply refer to a valid serialization, and avoid explicit reference to the monotone group when such is clear from context. We are now ready to provide an important definition.

Definition 3.1 (Implementation of Monotone Group) System $\mathbf{P}$ implements the monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ *if every execution* α *of* $\mathbf{P}$ *has a serialization that is valid for the monotone group.*

Denote $\mathsf{RMW}(\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle)$ the Read&Modify&Write operation associated with the monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ in the natural way. So, in particular, $\mathsf{RMW}(\langle \mathrm{Z}, \mathbb{N} \setminus \{0\}, +, \leq \rangle)$ and $\mathsf{RMW}(\langle \mathrm{Q}, \mathbb{N} \setminus \{0, 1\}, \cdot, \leq \rangle)$ are the RMW operations Fetch&Add and Fetch&Multiply, respectively. *Monotone* RMW operations are those associated with monotone groups. Say that system $\mathbf{P}$ implements the (monotone) operation $\mathsf{RMW}(\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle)$ whenever it implements the associated monotone group.

In our later definitions and proofs, we will sometimes write $\mathsf{In}_\alpha(op)$ and $\mathsf{Out}_\alpha(op)$ in order to emphasize reference to execution α.

We conclude this section with an immediate consequence of the *Valid Start* and *Valid Composition* conditions assumed in Definition **??**.

Property 3.1 *Assume that system* $\mathbf{P}$ *implements the monotone group* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$*. Then, for any operation* op *in an execution* α *of* $\mathbf{P}$*,*

$$\mathsf{Out}_\alpha(op) \;=\; \biguplus_{\|\{op' \;\mid\; op' \xrightarrow{\alpha} op\}\|} \left\{ \mathsf{In}_\alpha(op') \;\mid\; op' \xrightarrow{\alpha} op \right\} .$$

Our particular definitions for sequential consistency and linearizability will refer to any arbitrary execution α of the system $\mathbf{P}$, and to its valid serialization $S(\alpha)$.

Say that *a process* p_i *is sequentially consistent in execution* α [**?**] if the serialization $S(\alpha)$ extends $\xrightarrow{\alpha}_i$; that is, for any pair of operations $op_i^{(1)}$ and $op_i^{(2)}$ such that $op_i^{(1)} \xrightarrow{\alpha}_i op_i^{(2)}$, $op_i^{(1)} \xrightarrow{S(\alpha)} op_i^{(2)}$. The *Valid Composition* condition implies that for any two operations $op_i^{(1)}$ and $op_i^{(2)}$ such that $op_i^{(1)} \xrightarrow{S(\alpha)} op_i^{(2)}$, $\mathsf{Out}(op_i^{(1)}) \prec \mathsf{Out}(op_i^{(2)})$. Thus, it follows that for any process p_i that is sequentially consistent in execution α, for any pair of operations $op_i^{(1)}$ and $op_i^{(2)}$ such that $op_i^{(1)} \xrightarrow{\alpha} op_i^{(2)}$, $\mathsf{Out}(op_i^{(1)}) \prec \mathsf{Out}(op_i^{(2)})$.

Say that operation $op_i^{(1)}$ at process p_i in execution α is *sequentially inconsistent in execution* α if there is another operation $op_i^{(2)}$ at the same process in execution α such that $op_i^{(2)} \xrightarrow{\alpha} op_i^{(1)}$ while $op_i^{(2)} \xrightarrow{S(\alpha)} op_i^{(1)}$. Say that operation $op_i^{(1)}$ at process p_i in execution α is *sequentially consistent in execution* α if it is not sequentially inconsistent in execution α. Clearly, process p_i is sequentially consistent in execution α if every operation op_i at process p_i in execution α is sequentially consistent in it.

Say that execution α is *sequentially consistent* [**?**] if every process p_i is sequentially consistent in α. It follows that execution α is sequentially consistent if every operation in execution α is sequentially consistent in it. Finally, say that the system $\mathbf{P}$ is *sequentially consistent* if all its executions are.

Say that execution α is *linearizable* [**?**] if the serialization $S(\alpha)$ extends $\xrightarrow{\alpha}$; that is, for any pair of operations $op^{(1)}$ and $op^{(2)}$ such that $op^{(1)} \xrightarrow{\alpha} op^{(2)}$, $op^{(1)} \xrightarrow{S(\alpha)} op^{(2)}$. The *Valid Composition* condition implies that for any two operations $op^{(1)}$ and $op^{(2)}$ such that $op^{(1)} \xrightarrow{S(\alpha)} op^{(2)}$, $\mathsf{Out}(op^{(1)}) \prec \mathsf{Out}(op^{(2)})$. Thus, it follows that for any pair of operations $op^{(1)}$ and $op^{(2)}$ such that $op^{(1)} \xrightarrow{\alpha} op^{(2)}$, $\mathsf{Out}(op^{(1)}) \prec \mathsf{Out}(op^{(2)})$. Finally, say that system $\mathbf{P}$ is *linearizable* if all its executions are.

4 The Monotone Linearizability Lemma

Throughout this section, we refer to a distributed system $\mathbf{P}$ implementing a monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$. The *Monotone Linearizability Lemma* [**?**, Proposition 5.1] establishes ordering constraints of linearizability on the system $\mathbf{P}$. However, in order to provide the reader with useful intuition for the *Monotone Linearizability Lemma* and its proof, we first prove a corresponding *Monotone Sequential Consistency Lemma,* which establishes similar ordering constraints of sequential consistency on the system $\mathbf{P}$. The proof of the *Monotone Sequential Consistency Lemma* is substantially simpler and more succinct than the one of the *Monotone Linearizability Lemma.*

Recall that, by Lemma **??**, the monotone group $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ is pairwise independent. So, there are two distinct elements $a_1, a_2 \in \mathbb{M}$, with $a_1, a_2 \neq e$, that are pairwise independent over $\langle \mathbb{\Gamma}, \oplus \rangle$. The proof of the *Monotone Sequential Consistency Lemma* amounts to establishing a contradiction to pairwise independence for a hypothetical *non-sequentially consistent* execution, in which the types of the RMW operations issued by the processes are a_1 and a_2. We are now ready to state and prove the *Monotone Sequential Consistency Lemma.*

Proposition 4.1 (Monotone Sequential Consistency Lemma) *Consider an execution α of system $\mathbf{P}$ in which process p_1 issues only operations of type a_1, while any other process p_i, $i \neq 1$, issues only operations of type a_2. Then, p_1 is sequentially consistent in execution α.*

Proof: We start with an informal outline of our proof. We will proceed by contradiction. So, we will consider the earliest sequentially inconsistent operation $op_1^{(1)}$ at process p_1, and the latest operation $op_1^{(1)}$ (at p_1) that precedes it. We will use these operations to construct two executions γ_1 and γ_2 that are *indistinguishable* to process p_1 with respect to operation $op_1^{(2)}$. This indistinguishability implies that $op_1^{(2)}$ receives the same output in these two executions. Then, the contradiction will follow from the comparison of the two identical outputs. where we use simple algebraic properties of (monotone) groups in order to contradict the assumed pairwise independence. We now continue with the details of the formal proof.

Assume, by way of contradiction, that process p_1 is *not* sequentially consistent in execution α. So, there is at least one operation at p_1 that is sequentially inconsistent in execution α. Consider the *earliest* such operation $op_1^{(1)}$, and let $op_1^{(2)}$ be the *latest* operation at process p_1 that precedes $op_1^{(1)}$ in α. So, $op_1^{(2)} \xrightarrow{\alpha} op_1^{(1)}$ while $op_1^{(1)} \xrightarrow{S(\alpha)} op_1^{(2)}$, where $S(\alpha)$ is the (unique) valid serialization of α.

Denote $k \geq 0$ the number of operations at p_1 that precede $op_1^{(2)}$ in execution α. Since all these operations are sequentially consistent in execution α, they precede $op_1^{(2)}$ in $S(\alpha)$ as well.

In our proof, we will use the operations $op_1^{(1)}$ and $op_1^{(2)}$ in order to define and treat two finite prefixes of execution α:

- the finite prefix β_1 of execution α that ends with the response for operation $op_1^{(1)}$, and
- the finite prefix β_2 of execution α that ends with the response for operation $op_1^{(2)}$.

Clearly, β_2 is a prefix of β_1 as well. We first treat separately each of the two prefixes β_1 and β_2, and its corresponding extension; we then treat them together.

Properties of the prefix β_1 and its extension γ_1:

Consider a finite execution γ_1, which is an extension of β_1 that includes no additional invocations by processes; so, γ_1 is an extension of β_1 that additionally includes only responses to invocations that are pending in β_1.

Since β_1 is a prefix of both α and γ_1, it follows that all operations whose responses are included in β_1 (or, in other words, they are not preceded in either α or γ_1 by the response for $op_1^{(1)}$) have identical outputs in α and γ_1. In particular, $\mathsf{Out}_\alpha\left(op_1^{(1)}\right) = \mathsf{Out}_{\gamma_1}\left(op_1^{(1)}\right)$ and $\mathsf{Out}_\alpha\left(op_1^{(2)}\right) = \mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right)$. Take now the (unique) valid serialization $S(\gamma_1)$ of γ_1.

Since $op_1^{(1)} \xrightarrow{S(\alpha)} op_1^{(2)}$, the *Valid Composition* condition (for $S(\alpha)$) implies that $\mathsf{Out}_\alpha\left(op_1^{(1)}\right) \prec \mathsf{Out}_\alpha\left(op_1^{(2)}\right)$. Since $\mathsf{Out}_\alpha\left(op_1^{(1)}\right) = \mathsf{Out}_{\gamma_1}\left(op_1^{(1)}\right)$ and $\mathsf{Out}_\alpha\left(op_1^{(2)}\right) = \mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right)$, it follows that $\mathsf{Out}_{\gamma_1}\left(op_1^{(1)}\right) \prec \mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right)$. The *Valid Composition* condition (for $S(\gamma_1)$) implies now that $op_1^{(1)} \xrightarrow{S(\gamma_1)} op_1^{(2)}$,

Since the outputs of the k operations (of type a_1) at p_1 that precede $op_1^{(2)}$ in execution α. are identical in α and γ_1, the *Valid Composition* condition (for γ_1) implies that all these operations precede $op_1^{(2)}$ in $S(\gamma_1)$ as well.

Denote l_1 the number of operations at other processes that precede $op_1^{(2)}$ in the serialization $S(\gamma_1)$; recall that all these operations are of type a_2.

By Property **??**, $\mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right)$ is a composite expression involving $k+1$ contributions of a_1 and l_1 contributions of a_2. By the *Commutativity* property, these two types of contributions can be separated from each other in the composite expression, so that

$$\mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right) \quad = \quad \bigoplus_{k+1} a_1 \oplus \bigoplus_{l_1} a_2\,.$$

Properties of the prefix β_2 and its extension γ_2:

Consider a finite execution γ_2, which is an extension of β_2 that includes no additional invocations by processes; so, γ_2 is an extension that only includes responses to invocations that are pending in β_2.

Since β_2 is a prefix of both α and γ_2, it follows that all operations whose responses are included in β_2 (hence, they are not preceded in either α or γ_2 by the response for $op_1^{(2)}$) have identical outputs in α and γ_2. In particular, $\mathsf{Out}_\alpha\left(op_1^{(2)}\right) = \mathsf{Out}_{\gamma_2}\left(op_1^{(2)}\right)$. Take now the (unique) valid serialization $S(\gamma_2)$ of execution γ_2.

Since the outputs of the k operations (of type a_1) at p_1 that precede $op_1^{(2)}$ in execution α. are identical in α and γ_2, the *Valid Composition* condition (for γ_2) implies that all these operations precede $op_1^{(2)}$ in $S(\gamma_2)$ as well.

Denote l_2 the number of operations (all of type a_2) at other processes that precede $op_1^{(2)}$ in the serialization $S(\gamma_2)$.

By Property **??**, $\mathsf{Out}_{\gamma_2}\left(op_1^{(2)}\right)$ is a composite expression involving k contributions of a_1 and l_2 contributions of a_2. By the *Commutativity* property, these two types of contributions can be separated from each other in the composite expression, so that

$$\mathsf{Out}_{\gamma_2}\left(op_1^{(2)}\right) \quad = \quad \bigoplus_{k} a_1 \oplus \bigoplus_{l_2} a_2 \,.$$

<u>**Joint properties of the prefixes β_1 and β_2 and their extensions γ_1 and γ_2:**</u>
Recall that $\mathsf{Out}_{\alpha}\left(op_1^{(2)}\right) = \mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right)$ and $\mathsf{Out}_{\alpha}\left(op_1^{(2)}\right) = \mathsf{Out}_{\gamma_2}\left(op_1^{(2)}\right)$; hence, it follows that $\mathsf{Out}_{\gamma_1}\left(op_1^{(2)}\right) = \mathsf{Out}_{\gamma_2}\left(op_1^{(2)}\right)$, so that

$$\bigoplus_{k+1} a_1 \oplus \bigoplus_{l_1} a_2 \quad = \quad \bigoplus_{k} a_1 \oplus \bigoplus_{l_2} a_2 \,.$$

By Property **??**, $\bigoplus_{l_2} a_2 = \bigoplus_{l_2 - l_1} a_2 \oplus \bigoplus_{l_1} a_2$. Hence, by Property **??**, it follows that $a_1 = \bigoplus_{l_2 - l_1} a_2$. So, a_1 and a_2 are *not* pairwise independent over $\langle \mathbb{I\Gamma}, \oplus \rangle$. A contradiction. ■

We now return to the Monotone Linearizability Lemma. Recall that, by Lemma **??**, the monotone group $\langle \mathbb{I\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$ is n-wise independent for any integer $n \geq 2$. So, there are n distinct elements $a_1, a_2, \ldots, a_n \in \mathbb{M}$, with $a_1, a_2, \ldots, a_n \neq e$, which are n-wise independent over $\langle \mathbb{I\Gamma}, \oplus \rangle$. The proof of the *Monotone Linearizability Lemma* amounts to establishing a contradiction to n-wise independence for a hypothetical *non-linearizable* execution, in which the types of the RMW operations issued by the processes are $a_1, a_2, \ldots, a_n$.

Proposition 4.2 (Monotone Linearizability Lemma) *Consider any execution α of system $\mathbf{P}$ in which each process p_i issues only operations of type a_i, where $1 \leq i \leq n$. Then, α is linearizable.*

5 Application to Switching Networks

A *switching network* [**?**] is a directed, acyclic graph made up of nodes called *switches* and *output registers,* and edges called *wires.* Whenever a process issues a RMW operation, it shepherds a *token* through the network, which traverses a path of switches. Both switches and tokens have internal states. A token arrives at a switch via an *input wire.* In a single atomic step, the switch and the token change their states, and the token leaves the switch on an *output wire.* The token is eventually returned a value when it arrives at an output register.

Clearly, concurrent processes are spatially dispersed in a switching network, which reduces their simultaneous crossings in front of the same memory location (switch or output register). This offers potential for low contention.

The *size* of a switching network is the total number of switches in it; its *latency* is the maximum number of switches traversed by a token shepherding a RMW operation through the network. Thus, size and latency are two natural measures for space complexity and time complexity, respectively, in the model of switching networks.

The *concurrency* of a switching network is the maximum number of concurrent *processes* that may simultaneously shepherd a RMW operation through the network.

In order to model the low-contention property for switching networks, Busch *et al.* [**?**] introduced *register bottleneck* and *layer bottleneck*; roughly speaking, both register bottleneck and layer bottleneck measure the *minimum* number of network elements (either switches or output registers) that are accessed by processes in any infinite execution. (Layer bottleneck assumes partitioning the switches of the network into *layers* in the natural way.) Intuitively, if this minimum number is small, some network element will become a *bottleneck* (or a *"hot-spot"* in the pool of memory locations) in some infinite execution and the network incurs high contention; hence, a switching network is *low-contention* if register bottleneck and layer bottleneck are sufficiently large.

For switching networks with switches with a finite number of states, Busch *et al.* [**?**, Theorem 6.1] prove:

Theorem 5.1 (Impossibility Result for Finite-Switch Networks) *There exists no non-trivial, finite-switch switching network* $\mathcal{N}$ *with concurrency* $(or(\mathcal{N})+1)\cdot\left(S^{size(\mathcal{N})}+1\right)$ *that has finite size, incurs register bottleneck at least* 2 *and implements a monotone group* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$.

For switching networks with switches with an infinite number of states, Busch *et al.* [**?**, Theorem 6.2] prove:

Theorem 5.2 (Impossibility Result for Infinite-Switch Networks) *There exists no non-trivial, infinite-switch switching network with unbounded concurrency that has finite size, incurs layer bottleneck at least* 2 *and implements a monotone group* $\langle \mathbb{\Gamma}, \mathbb{M}, \oplus, \preceq \rangle$.

6 Epilogue

We surveyed an algebraic approach toward proving impossibility results in distributed computing, based on a new class of algebraic groups, called *monotone groups,* recently introduced by Busch *et al.* [**?**]. The approach is both interesting and promising. We would like to use monotone groups for investigating the possibility of implementing other, *non-monotone* RMW operations with finite-sized, highly concurrent, low-contention switching networks, or in other (than switching networks) models of distributed computing.

References

[1] C. Busch, M. Mavronicolas and P. Spirakis, "The Cost of Concurrent, Low-Contention Read&Modift&Write, *Theoretical Computer Science,* Vol. 333, No. 3, pp. 373–400, March 2005.

[2] P. Fatourou and M. Herlihy, "Read-Modify-Write Networks," *Distributed Computing,* Vol. 17, pp. 33–46, 2004.

[3] A. M. W. Glass, *Partially Ordered Groups,* Vol. 7, Series in Algebra, World Scientific, 1999.

[4] M. Hall, Jr., *Theory of Groups,* Second Edition, Chelsea Publishing Company, 1979.

[5] M. Herlihy and J. Wing, "Linearizability: A Correctness Condition for Concurrent Objects," *ACM Transactions on Programming Languages and Systems,* Vol. 12, No. 3, pp. 463–492, July 1990.

[6] M. Klugerman and C. G. Plaxton, "Small-Depth Counting Networks," *Proceedings of the 24th Annual ACM Symposium on Theory of Computing,* pp. 417–428, May 1992.

[7] L. Lamport, "How to Make a Multiprocessor Computer that Correctly Executes Multiprocess Programs," *IEEE Transactions on Computers,* Vol. C-28, No. 9, pp. 690–691, September 1979.