Article

Analysis of a 4-Bit Feedback Shift Register $f(x_1, x_2, x_3, x_4) = 1 \oplus x_2(x_1 \vee x_4)$

Makhabbat Batyrova* [1] and Alikhan Zhazaibek [1]

[1]Department of Mathematics and Natural Sciences, SDU University, Kaskelen, Kazakhstan

DOI: 10.47344/b3btd773

Abstract

We study a 4-bit feedback shift register (FSR) whose Boolean feedback function is $f(x_1, x_2, x_3, x_4) = 1 \oplus x_2(x_1 \vee x_4)$. We enumerate transitions from all 16 initial states, determine singularity, identify cycles, compute the ultimate output periodicity and derive the Algebraic Normal Form (ANF) of the feedback function. The FSR is shown to be singular, possessing a unique 3-cycle that acts as a global attractor for all states. Through algebraic analysis, we determine that the feedback function has an algebraic degree of 3, which provides high non-linearity but fails to overcome the structural weakness of the state-transition bijectivity collapse. Furthermore, we analyze the hardware implementation complexity and the topological structure of the transient trees. A Python program is provided that reproduces the full state enumeration algorithmically and presents results in compact tables. All 16 states are shown to converge to the single cycle $(0110 \to 1011 \to 1101 \to 0110)$, yielding a maximal period of 3. These findings have direct relevance to the cryptanalysis of stream ciphers based on non-linear feedback shift registers (NLFSRs).

Keywords: feedback shift register, Boolean feedback, singular FSR, state graph, period, algebraic normal form, hardware complexity

I. Introduction

Feedback shift registers (FSRs) are fundamental building blocks in modern digital communications. A central concern in their analysis is the periodicity of output sequences, the bijectivity of the state-transition map and the algebraic properties of the feedback function as these characteristics directly influence both cryptographic strength and practical usability.

A *linear* FSR (LFSR), whose feedback function is restricted to a linear Boolean function (typically constructed using only XOR gates), is highly mathematically tractable: assuming a standard non-singular configuration where the oldest bit is tapped, its state-transition map is bijective, sequences are always periodic and maximum-length sequences (m-sequences) achieve the maximal period $2^L - 1$ [1], [4]. However, LFSRs are notoriously vulnerable to algebraic cryptanalysis (see, e.g. [2]).

In order to tackle such problems, modern cryptographic designs turn to *nonlinear* FSRs (NLFSRs). NLFSRs admit richer and exponentially more complex behavior. Nevertheless, this complexity is a double-edged sword: without careful design, the transition

*Corresponding author: 230183099@sdu.edu.kz
Email: 230183099@sdu.edu.kz ORCID: 0009-0006-8601-878X
Email: 230183110@sdu.edu.kz ORCID: 0009-0009-7979-810X

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

map may become non-injective (singular). Singular NLFSRs exhibit fatal flaws for pseudorandom generation, including transient states, multiple disconnected cycle lengths, state collapse and irregular output patterns [2], [6].

The present paper provides analysis of a specific 4-bit NLFSR whose feedback function is

$$f(x_1, x_2, x_3, x_4) = 1 \oplus x_2\,(x_1 \vee x_4).$$

This is a nonlinear expression combining the logical operators XOR, AND, and OR. Despite its small state space, this register encapsulates all the characteristic phenomena of singular NLFSRs. Furthermore, it serves as an excellent case study for bridging software-based path tracing with theoretical algebraic properties.

The analysis is conducted exhaustively over all $2^4 = 16$ states. We derive the Algebraic Normal Form (ANF) to assess its cryptographic degree, prove singularity mathematically, determine the complete topological cycle structure of the state graph and compute both per-state periodicity and the ultimate output period. All theoretical results are cross-verified by a Python code that performs the full state enumeration algorithmically.

The rest of the paper is organized as follows. Section II outlines the necessary theoretical preliminaries. Section III provides an algebraic and hardware analysis of the feedback function. Section IV states and proves the main results regarding singularity and cycle topology. Section V details the software implementation. Section VI discusses the experimental results, including a comparative analysis with linear and de Bruijn models. Finally, Section VII concludes the paper.

II. Preliminaries

Let $\mathcal{S} = \mathbb{F}_2^4$ denote the set of all 4-bit states $x = (x_1, x_2, x_3, x_4)$ with $x_i \in \{0, 1\}$. Given a Boolean feedback function $f : \mathcal{S} \to \mathbb{F}_2$, one step of the FSR is defined as a right shift with the new bit $f(x)$ inserted on the left. The state transition map $T : \mathcal{S} \to \mathcal{S}$ is formulated as:

$$T(x_1, x_2, x_3, x_4) = \big(f(x_1, x_2, x_3, x_4),\ x_1,\ x_2,\ x_3\big). \tag{1}$$

Here, x_1 represents the newest bit entering the register (stage $L-1$), while x_4 represents the oldest bit (stage 0) that is discarded and forms the output sequence. In our specific architectural model, the feedback function is:

$$f(x_1, x_2, x_3, x_4) = 1 \oplus \big(x_2 \wedge (x_1 \vee x_4)\big),$$

where $\oplus$ denotes XOR (addition modulo 2), $\wedge$ denotes AND (multiplication modulo 2), and $\vee$ denotes logical OR.

Definition II.1 (Feedback Shift Register [2])**.** *A feedback shift register (FSR) of length L consists of L stages numbered $0, 1, \ldots, L-1$, each storing one bit. At each clock step: (1) the content of stage 0 is output; (2) stage i moves to stage $i-1$ for $1 \le i \le L-1$; (3) the new content of stage $L-1$ is evaluated via $s_j = f(s_{j-1}, \ldots, s_{j-L})$.*

Definition II.2 (Non-singular FSR [2])**.** *An FSR is* non-singular *if and only if the state-transition map $T : \mathcal{S} \to \mathcal{S}$ is bijective (a one-to-one and onto mapping), so that every state has an indegree and outdegree of exactly 1 in the directed state graph. If any state has an indegree of 0 or > 1, the FSR is* singular.

Definition II.3 (Periodicity and Tails [2])**.** *An initial state $x^{(0)} \in \mathcal{S}$ is* periodic *if $T^k(x^{(0)}) = x^{(0)}$ for some integer $k \ge 1$; the smallest such k is its* period*. Because the state space is finite, every forward orbit eventually enters a directed cycle. The length of this cycle is the* ultimate period*, and the number of state transitions required before entering the cycle is defined as the* tail length *(or transient phase).*

Definition II.4 (Algebraic Normal Form – ANF [7])**.** *Every Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be uniquely expressed as a multivariate polynomial over $\mathbb{F}_2$. This representation is called the Algebraic Normal Form (ANF) and takes the shape: $f(x_1, \ldots, x_n) = a_0 \oplus \bigoplus_{i=1}^{n} a_i x_i \oplus \bigoplus_{1 \le i < j \le n} a_{i,j} x_i x_j \oplus \cdots \oplus a_{1,2,\ldots,n} x_1 x_2 \ldots x_n$, where $a \in \{0, 1\}$.*

Definition II.5 (Algebraic Degree [7])**.** *The* algebraic degree*, denoted as $deg(f)$, is the number of variables in the highest-order non-zero term in the ANF of f. A function with degree 1 is linear or affine.*

Definition II.6 (de Bruijn FSR [1])**.** *A non-singular FSR of length L is a* de Bruijn FSR *if the state transition graph consists of a single cycle of length 2^L. Its output sequence is a* de Bruijn sequence*, which means every possible length-L substring occurs exactly once per period.*

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

III. Algebraic and Hardware Analysis

Before analyzing the state transitions, it is highly instructive to analyze the mathematical properties of the feedback function itself. The logical formulation $f = 1 \oplus x_2(x_1 \vee x_4)$ utilizes a mix of Boolean logic gates. For cryptographic analysis, this must be converted into the field $\mathbb{F}_2$.

A. Derivation of the Algebraic Normal Form

To find the ANF of the function (see Definition II.4), we must replace the logical OR operation ($\vee$) with its modulo-2 arithmetic equivalent. In $\mathbb{F}_2$, the OR operation is defined as:

$$A \vee B = A \oplus B \oplus (A \wedge B) \tag{2}$$

Applying identity (2) to the expression $(x_1 \vee x_4)$, we obtain:

$$x_1 \vee x_4 = x_1 \oplus x_4 \oplus x_1x_4 \tag{3}$$

Substituting (3) back into the original feedback function:

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= 1 \oplus \big(x_2 \wedge (x_1 \oplus x_4 \oplus x_1x_4)\big) \\ &= 1 \oplus (x_2x_1 \oplus x_2x_4 \oplus x_2x_1x_4) \\ &= 1 \oplus x_1x_2 \oplus x_2x_4 \oplus x_1x_2x_4 \end{aligned} \tag{4}$$

Equation (4) represents the canonical ANF of the feedback shift register.

Proposition III.1. *The algebraic degree of the feedback function is $deg(f) = 3$.*

Proof. The term with the maximum number of variables in the ANF (4) is $x_1x_2x_4$. Because this term consists of 3 distinct variables multiplied together (ANDed), the algebraic degree of the function is strictly 3 (see Definition II.5). □

In cryptographic contexts, a high algebraic degree is generally desired to protect against the Berlekamp-Massey algorithm and higher-order differential attacks [6]. For an n-bit register, the maximum possible degree is n. Thus, a degree of 3 for a 4-bit register is considered cryptographically strong. However, as we will demonstrate in the state analysis of Section IV, this algebraic strength is entirely undermined by the register's singularity.

B. Hardware Implementation Complexity

From an engineering perspective, the implementation of this NLFSR requires consideration of its logic gate equivalent. Using the original formulation $f = 1 \oplus x_2(x_1 \vee x_4)$, the hardware circuit requires:

- One 2-input OR gate for $(x_1 \vee x_4)$.
- One 2-input AND gate to multiply the result by x_2.
- One 2-input XOR gate to add 1 (which practically serves as a logical NOT gate, meaning the final XOR gate can be replaced simply by an inverter on the output of the AND gate).

This makes the NLFSR marginally slower in standard-cell ASICs or discrete logic circuits compared to its linear counterparts, as the signal must propagate sequentially through the OR gate, then the AND gate, and finally the NOT/XOR inversion before the clock cycle can safely shift the register. Note, however, that in modern FPGA architectures, any Boolean function of up to 4 variables maps to a single Look-Up Table (LUT). Therefore, the critical path delay and logic footprint in an FPGA will be completely identical for both this NLFSR and a standard 4-bit LFSR.

IV. Main Results: State Topology

A. Proof of Singularity

Theorem IV.1. *The NLFSR defined by* (1) *with $f = 1 \oplus x_2(x_1 \vee x_4)$ is singular.*

Proof. By exhaustive mathematical enumeration of all 16 states (detailed further in Table I), the transition map T creates severe clustering. Specifically, the states $0000, 0001, 0100, 0101, 1110, 1111$ are never generated as outputs of the transition function. Because their indegree is 0, they are *Garden-of-Eden* states. Conversely, states such as $0110, 0111, 1000, 1001, 1100, 1101$ each possess an indegree of 2, meaning two different internal states converge into the exact same subsequent state. Because the mapping is not $1:1$, the transition matrix is non-invertible, making T non-bijective. Therefore, the FSR is singular by Definition II.2. □

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

B. Cycle Structure and Attractors

Theorem IV.2. *The directed state graph has a single, unique* 3*-cycle*

$$\mathcal{C} = \big(0110 \ \rightarrow \ 1011 \ \rightarrow \ 1101 \ \rightarrow \ 0110\big).$$

This cycle acts as a global attractor. All 16 *states flow into this cycle, forming transient tree structures of varying depths.*

Proof. By analyzing the preimages of the states within the cycle, we can reconstruct the State Transition Graph (STG) topology. The cycle states are $S_c = \{0110, 1011, 1101\}$.

- The state 0110 has preimages $\{1100, 1101\}$. Since $1101 \in S_c$, the state 1100 is the root of a transient tree entering the cycle at 0110.
- Tracing backward from 1100, its preimages are $\{1000, 1001\}$.
- Tracing backward from 1000, its preimages are $\{0000, 0001\}$ (both Garden-of-Eden states, tail length 3).
- Tracing backward from 1001, its preimages are $\{0010, 0011\}$.
- State 0010 has the preimage 0101 (tail length 4).
- State 0011 has the preimage 0111, which in turn has preimages $\{1110, 1111\}$ (tail length 5).
- The cycle state 1101 has the transient preimage 1010. Tracing backward from 1010, its preimage is 0100 (a Garden-of-Eden state, tail length 2).

Because every recursive backward trace terminates at a Garden-of-Eden state, and all 16 states are accounted for within these traces, the STG contains no other disjoint cycles. The entire space collapses into $\mathcal{C}$. □

Corollary IV.1. *The maximal period of the FSR is* 3*, achieved exactly by the initial states* $\{0110, 1011, 1101\}$*. Every initial state has an ultimate output sequence period of* 3 *after a finite transient phase (maximum tail length of 5).*

V. Methods and Software Implementation

A. State Enumeration Algorithm

While theoretical preimage tracing (as in the proof of Theorem IV.2) proves the topology, software-based enumeration provides robust verification and scales effortlessly. The analysis proceeds by exhaustive traversal of the state graph. For each of the 16 initial states, the sequence of transitions under T (Equation 1) is computed iteratively until a repeated state is detected (cycle closure).

The indices of first visits are tracked in a hash map (dictionary) so that both the tail length and the cycle length can be read off in $O(1)$ lookup time. Singularity is independently tested by computing the indegree of every node: each state pushes one outgoing edge, and the resulting in-degree distribution array is inspected. Any value other than 1 immediately flags the FSR as singular (per Definition II.2).

B. Python Implementation

The following Python program implements the above procedures. It enumerates the full state graph, detects cycles and tails, prints per-state trajectories, and reports singularity and maximal period. Note that the program tracks the oldest bit (x_4) falling off the register to correctly represent the FSR's true output stream.

```
def feedback(x1, x2, x3, x4):
    # f = 1 XOR (x2 AND (x1 OR x4))
    return 1 ^ (x2 & (x1 | x4))

def step(state):
    x1, x2, x3, x4 = state
    f = feedback(x1, x2, x3, x4)
    return (f, x1, x2, x3), x4

def run_from(initial):
    seen_index = {initial: 0}
```

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

```
    states =[initial]
    outs =[]
    state = initial
    k = 0
    while True:
        state, out_bit = step(state)
        outs.append(out_bit)
        states.append(state)
        k += 1
        if state in seen_index:
            cycle_start = seen_index[state]
            cycle_len = k - cycle_start
            periodic = (state == initial)
            period = cycle_len if periodic else 0
            return {
                'initial': initial, 'states': states,
                'outs': outs, 'periodic': periodic,
                'period': period, 'cycle_len': cycle_len,
                'tail_len': cycle_start
            }
        else:
            seen_index[state] = k

def is_singular_all_states():
    states =[tuple(map(int, f'{i:04b}')) for i in range(16)]
    indeg = {s: 0 for s in states}
    for s in states:
        ns, _ = step(s)
        indeg[ns] += 1
    return not all(indeg[s] == 1 for s in states)

def state_to_str(state):
    return ''.join(str(b) for b in state)

all_states =[tuple(map(int, f'{i:04b}')) for i in range(16)]
results = [run_from(s) for s in all_states]

for r in results:
    outs_str = ''.join(str(b) for b in r['outs'])
    print(f'Start: {state_to_str(r["initial"])}  '
          f'Tail: {r["tail_len"]}  Cycle: {r["cycle_len"]}  '
          f'Periodic: {r["periodic"]}  Period: {r["period"]}  '
          f'First outputs: {outs_str}')

singular = is_singular_all_states()
max_period = max(r['period'] for r in results)
```

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

```
max_period_starts = [state_to_str(r['initial']) for r in results
                     if r['period'] == max_period and max_period > 0]

print(f'FSR singular: {singular}')
print(f'Maximal period: {max_period}  '
      f'(achieved by: {", ".join(max_period_starts)})')
```

C. Complexity and Performance Assessment

Time Complexity. For an n-bit register, there are 2^n initial states. The provided path-tracing algorithm runs in $O(2^{2n})$ worst-case time, as orbits are traced independently for each initial state without a global memoization matrix linking separate starting nodes. For $n = 4$, $2^{2(4)} = 256$ operations, which executes practically instantaneously.

Space Complexity. Memory utilization scales at $O(2^n)$ to hold the 'seen_index' hash map for the longest possible un-branched path. This is exceptionally efficient and ensures that the codebase can scale to test slightly larger registers (e.g., $n = 16, 24$) before combinatorial explosion requires transitioning to C++ or parallel GPU computing.

VI. Results and Discussion

A. Transition Table Analysis

Table I lists the one-step transition for all 16 states alongside the computed feedback bit, validating the software output against the mathematical singularity proof of Theorem IV.1.

TABLE I
One-Step Transitions for All 16 States

State $(x_1x_2x_3x_4)$	**Feedback bit** f	**Next state** $T(x)$
0000	1	1000
0001	1	1000
0010	1	1001
0011	1	1001
0100	1	1010
0101	0	0010
0110	1	1011
0111	0	0011
1000	1	1100
1001	1	1100
1010	1	1101
1011	1	1101
1100	0	0110
1101	0	0110
1110	0	0111
1111	0	0111

The empirical mapping clearly demonstrates non-injectivity. For instance, the transition function evaluates to 1000 for inputs 0000 and 0001, fundamentally deleting the uniqueness of the history of the register. This proves that time-reversal on this FSR is mathematically impossible, as entropy is destroyed at the merge points.

B. Per-State Periodicity Summary

Table II summarises the exact topological metrics for each individual starting state, noting the distance to the attractor cycle (Tail) and the local periodicity. The maximum tail length of 5 and the uniform cycle length of 3 confirm Corollary IV.1.

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

TABLE II
PER-STATE SUMMARY: TAIL, CYCLE, PERIODICITY, AND FIRST OUTPUTS

Init	Tail	Cycle	Periodic?	Period	First outputs
0000	3	3	No	–	000011
0001	3	3	No	–	100011
0010	3	3	No	–	010011
0011	3	3	No	–	110011
0100	2	3	No	–	00101
0101	4	3	No	–	1010011
0110	0	3	Yes	3	011
0111	4	3	No	–	1110011
1000	2	3	No	–	00011
1001	2	3	No	–	10011
1010	1	3	No	–	0101
1011	0	3	Yes	3	110
1100	1	3	No	–	0011
1101	0	3	Yes	3	101
1110	5	3	No	–	01110011
1111	5	3	No	–	11110011

C. Comparative FSR Analysis

To fully contextualize the weakness of this singular NLFSR, we provide a comparative analysis against two other standard 4-bit models in Table III: a primitive Linear FSR ($f = x_1 \oplus x_4$) and an optimal de Bruijn sequence generator (Definition II.6).

TABLE III
COMPARISON OF 4-BIT FSR ARCHITECTURES

Metric	Proposed Singular NLFSR	Primitive LFSR	de Bruijn NLFSR
Feedback Function	$1 \oplus x_2(x_1 \vee x_4)$	$x_1 \oplus x_4$	$x_1 \oplus x_4 \oplus (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$
Algebraic Degree	3	1 (Linear)	3 (Nonlinear)
Max Output Period	3	15 ($2^n - 1$)	16 (2^n)
Bijectivity	Singular (Non-invertible)	Non-singular (Bijective)	Non-singular (Bijective)
Garden-of-Eden States	6	0	0
Attractors	Single 3-cycle	15-cycle, 1-cycle (Zero)	Single 16-cycle
Cryptographic Utility	Unusable	Weak (Berlekamp-Massey)	High (Stream Ciphers)

D. Discussion on Cryptographic Viability

As detailed in Table III, while the proposed FSR matches the high algebraic degree (degree 3) of the optimal de Bruijn FSR and strictly exceeds that of the primitive LFSR, it fails completely on a macroscopic structural level.

The singular nature of the transition map (Theorem IV.1) creates a cascading state-collapse. From a cryptographic standpoint, this singularity is disastrous. It indicates that the key space (the initial state) shrinks exponentially as the register is clocked. Regardless of which of the 16 states the user inputs as a secret key, the attacker only needs to analyze the output corresponding to the 3-cycle identified in Theorem IV.2. An attacker observing the output sequence will see repetitions of “011”, “110”, or “101” almost immediately.

This analysis underscores a critical paradigm in stream cipher design: local non-linearity in the feedback formula does not guarantee global randomness [5]. Maximizing the period to 2^L (the de Bruijn property, Definition II.6) requires strict adherence to cycle-joining theorems, ensuring that the transition map remains perfectly bijective while incorporating the nonlinear terms. The

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.

presence of the AND and OR gates in our studied register created an asymmetry that favored zero-generation over balanced bit propagation, forcing the state space to fold into itself.

VII. Conclusion and Future Work

We have provided a comprehensive analytical and empirical evaluation of a 4-bit feedback shift register governed by the feedback function $f = 1 \oplus x_2(x_1 \vee x_4)$. By transforming the logical function into its Algebraic Normal Form (Section III-A), we identified its cryptographic degree to be 3 (Proposition III.1). Despite this high non-linearity, the exhaustive state enumeration mathematically proved the register to be *singular* (Theorem IV.1), exhibiting severe transition map non-injectivity. The state space collapses entirely, featuring six Garden-of-Eden states and converging into a single attracting 3-cycle ($0110 \rightarrow 1011 \rightarrow 1101 \rightarrow 0110$) with a maximum transient tail of 5 steps (Theorem IV.2, Corollary IV.1).

Because the maximal output period is strictly capped at 3, well below the theoretical maximum of 16, these results conclusively rule out the use of this specific register configuration as a standalone pseudorandom number generator, keystream generator or error-correcting code sequencer.

Future work could expand this analysis framework to 8-bit and 16-bit architectures. A highly automated algorithmic sweep over all Boolean functions of a given algebraic degree could be utilized to map the exact mathematical boundaries separating singular FSRs from non-singular de Bruijn generators (Definition II.6). Additionally, studying compositions of these short-cycle singular FSRs could reveal whether their combined nonlinear outputs can theoretically reconstitute the bijectivity required for modern cryptographic standards.

Contributions

Makhabbat Batyrova: Responsible for the theoretical formulation and analysis of the feedback shift register model. Conducted the Algebraic Normal Form derivation, mathematically verified the singularity and state transition topology, and prepared the complete LaTeX typesetting of the manuscript.

Alikhan Zhazaibek: Developed, optimized, and implemented the Python algorithmic program for exhaustive state enumeration and cycle detection. Conducted computational experiments, organized numerical results, and compiled the per-state summary and comparative analysis tables.

Both authors jointly reviewed the outcomes, verified the correctness of the analytical logic, and contributed to the discussion and final drafting of the paper.

References

[1] S. W. Golomb, *Shift Register Sequences: Secure and Limited-Access Code Generators, Efficiency Code Generators, Prescribed Property Generators, Mathematical Models*, World Scientific, 2017.

[2] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996. Available: https://cacr.uwaterloo.ca/hac/

[3] A. Klein, *Stream Ciphers*, Springer, 2013. DOI: 10.1007/978-1-4471-5079-4.

[4] R. A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer, 2012 (Reprint of 1986 edition). DOI: 10.1007/978-3-642-82865-2.

[5] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*, Springer, 2010. DOI: 10.1007/978-3-642-04101-3.

[6] C. Ding, G. Xiao, and W. Shan, *The Stability Theory of Stream Ciphers*, Springer, 1991. DOI: 10.1007/3-540-54973-0.

[7] C. Carlet, *Boolean Functions for Cryptography and Coding Theory*, Cambridge University Press, 2020. DOI: 10.1017/9781108606806.

Received: October 21, 2025. Reviewed: December 9, 2025 - February 2, 2026. Accepted: February 20, 2026.