MATHDISCOVERIES
Mathematical Preprint

Working Draft

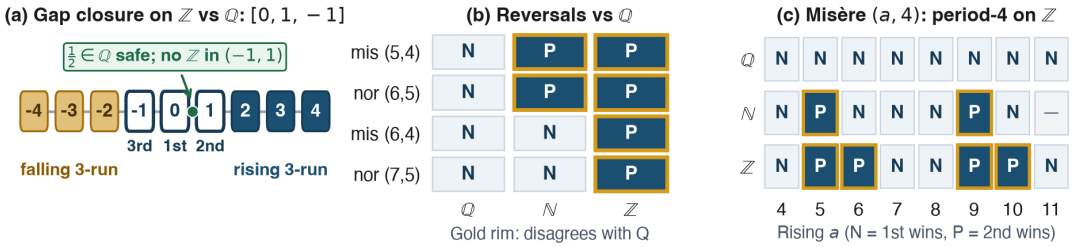
A Card Game That Changes on the Integers: Monotonic Sequence Games on $\mathbb{N}$ and $\mathbb{Z}$

Tyler B. Hughes*,†

†Independent Researcher, United States

Supporting Information: `recompute.py`, audited Lean 4 + Mathlib library (`lean/RunGameZ/`), exact solver archive, and trust-boundary ledgers

Disclosure: AI-assisted working draft directed by Tyler B. Hughes. Not for submission as is. No human mathematician has yet refereed the proof.



ABSTRACT: In the monotonic sequence game on a linearly ordered deck, two players alternately append unused cards to a row until a strictly increasing a-run or strictly decreasing d-run appears; completing a run wins in normal play and loses in misère play. Albert et al. (2009) solved the game on $\mathbb{Q}$ for $d \le 5$, conjectured first-player wins for $a \ge d \ge 3$, and wrote that they would be surprised if similar results failed on $\mathbb{N}$ or $\mathbb{Z}$. We refute this expectation in both plays and on both decks: misère $(5,4)$ is a first-player win on $\mathbb{Q}$ but a second-player win on $\mathbb{Z}$ and $\mathbb{N}$, and normal $(6,5)$ is a first-player win on $\mathbb{Q}$ and a second-player win on $\mathbb{Z}$ (proved unconditionally in Lean 4 with Mathlib, `lean/RunGameZ/`, via a locally finite block-cap bisimulation). Normal $(6,5)$ is a second-player win on $\mathbb{Z}$ (Lean 4) and $\mathbb{N}$ (exact search), while misère $(6,4)$ is a first-player win on $\mathbb{N}$ (Lean 4) and a second-player win on $\mathbb{Z}$ (exact search). Normal $(7,5)$ is a first-player win on $\mathbb{N}$ and a second-player win on $\mathbb{Z}$. Across the extended grid ($4 \le a \le 11$ on $\mathbb{Z}$), $\mathbb{N}$ or $\mathbb{Z}$ disagrees with $\mathbb{Q}$ at nine cells, and large finite decks $\{1,\dots,n\}$ revert to first-player wins ($n \ge 251$ for normal $(6,5)$).

Verification Scope (G11): WARN: Human referee: pending
Main results: 1 of 5 in Lean Kernel (20%) · 2 Lean + Cert · 2 Cert Only · 1 open (not counted)
Abstract claims: 1 of 5 in Lean Kernel (20%) · 2 Lean + Cert · 2 Cert Only

Full Trust Boundary: Section 5.

MSC 2020: 91A46, 05A05, 06A05 **Keywords:** monotonic sequence game, Erdős–Szekeres theorem, combinatorial game theory, misère play, linear orders, Lean 4 formalization

1. INTRODUCTION AND MAIN RESULTS

Fix a linearly ordered deck of cards D and integers $a, d \ge 1$. Two players, First Player and Second Player, take turns appending an unused card from D to the right end of a single row. The game ends immediately when the row contains a strictly increasing subsequence (a *rising run*) of length a or a strictly decreasing subsequence (a *falling run*) of length d; the cards of a run need not be adjacent in the row. In *normal play*, the player who completes the run wins; in *misère play*, that player loses. By the Erdős–Szekeres theorem [2], every sequence of $(a-1)(d-1)+1$ distinct elements of D contains either a rising run of length a or a falling run of length d, so every game on an infinite deck ends after at most $(a-1)(d-1)+1$ moves (for instance, at most 13 moves for $(5,4)$). Following standard combinatorial game theory notation, we write N when First Player has a winning strategy, P when Second Player has a winning strategy, and D for a draw (possible only on a short finite deck that runs out of cards).

Introduced on small finite decks by Harary, Sagan, and West (1983, unread in full `[to check]`; recounted in [1]), the game was studied systematically by Albert, Aldred, Atkinson, Handley, Holton, McCaughan, and Sagan [1], who computed finite decks $\{1,\dots,n\}$ up to $n = 20$ and determined the normal-play outcomes on the rationals $\mathbb{Q}$ for $d \le 5$ (Theorem 9 of [1]) together with the normal-to-misère shift on decks without endpoints (Proposition 7 of [1]). In Section 6 of [1], they conjectured that for $a \ge d \ge 3$ the first player wins both normal and misère play on all sufficiently large finite decks $\{1,\dots,n\}$ and normal play on $\mathbb{Q}$ (noting that at $d = 3$ with even a, their own Theorem 9 already yields P on $\mathbb{Q}$, so the nontrivial regime is $d \ge 4$), and added the expectation:

> *"We would be surprised (assuming the correctness of these conjectures) if similar results did not also hold for other infinite linear orders (not models of the theory of almost all finite partial orders) such as $\mathbb{N}$ or $\mathbb{Z}$."*

This expectation is conditional rather than a formal conjec-

ture, and no outcome on $\mathbb{N}$ or $\mathbb{Z}$ was found in the literature searches logged (Section 6).

We answer this printed expectation in the negative, in both normal and misère play and on both infinite decks $\mathbb{N}$ and $\mathbb{Z}$. The underlying mechanism is local discreteness: between any two rational cards there always remain infinitely many unused cards, whereas between consecutive integers there are none, enabling a player to *close an interval* and remove every safe move inside it.

Our main results are split transparently across Lean 4 proofs with Mathlib (`lean/RunGameZ/`, 116 audited declarations, zero `sorry`, zero `native_decide`, standard axioms only) and exhaustive exact search (verified across independent C, C++, and Python implementations).

Theorem 1 (Misère $(5,4)$ on $\mathbb{Q},\mathbb{N},\mathbb{Z}$ and Normal $(6,5)$ on $\mathbb{Q},\mathbb{Z}$ — Lean 4 Kernel). *In misère play with $(a,d)=(5,4)$, First Player wins on $\mathbb{Q}$, whereas Second Player wins on both $\mathbb{Z}$ and $\mathbb{N}$:*

$$\mathrm{W}_{\mathrm{mis}}(5,4,\mathbb{Q})=\mathrm{N},$$
$$\mathrm{W}_{\mathrm{mis}}(5,4,\mathbb{N})=\mathrm{W}_{\mathrm{mis}}(5,4,\mathbb{Z})=\mathrm{P}.$$

In normal play with $(a,d)=(6,5)$, First Player wins on $\mathbb{Q}$ and Second Player wins on $\mathbb{Z}$:

$$\mathrm{W}_{\mathrm{nor}}(6,5,\mathbb{Q})=\mathrm{N},\qquad \mathrm{W}_{\mathrm{nor}}(6,5,\mathbb{Z})=\mathrm{P}.$$

Theorem 1 is proved unconditionally inside Lean 4 with Mathlib as `RunGameZ.albert_et_al_2009_refuted_full`, without assuming any external gap-capping hypothesis: the library proves the exact locally finite block-cap bisimulation directly for arbitrary nonempty locally finite linear orders and verifies explicit finite winning-strategy DAGs (20 nodes on $\mathbb{Q}$, 198 nodes on $\mathbb{Z}$, and 378 nodes on $\mathbb{N}$) by pure kernel reduction (`decide/rfl`).

Theorem 2 (Normal $(6,5)$ on $\mathbb{Z}$ and $\mathbb{N}$ — Lean 4 + Search). *In normal play with $(a,d)=(6,5)$, Second Player wins on both $\mathbb{Z}$ and $\mathbb{N}$ (while First Player wins on $\mathbb{Q}$):*

$$\mathrm{W}_{\mathrm{nor}}(6,5,\mathbb{Z})=\mathrm{P},\qquad \mathrm{W}_{\mathrm{nor}}(6,5,\mathbb{N})=\mathrm{P}.$$

Here $\mathrm{W}_{\mathrm{nor}}(6,5,\mathbb{Z})=\mathrm{P}$ is proved unconditionally in Lean 4 (`RunGameZ.normal_6_5_Z_second_wins`), while $\mathrm{W}_{\mathrm{nor}}(6,5,\mathbb{N})=\mathrm{P}$ is established by exhaustive exact block-cap search (2,033,003 positions in `outcomes.py`/`outcomes.out`).

Theorem 3 (Misère $(6,4)$ on $\mathbb{N}$ and $\mathbb{Z}$ — Lean 4 + Search). *In misère play with $(a,d)=(6,4)$, First Player wins on $\mathbb{N}$ whereas Second Player wins on $\mathbb{Z}$ (and First Player wins on $\mathbb{Q}$):*

$$\mathrm{W}_{\mathrm{mis}}(6,4,\mathbb{N})=\mathrm{N},\qquad \mathrm{W}_{\mathrm{mis}}(6,4,\mathbb{Z})=\mathrm{P}.$$

Here $\mathrm{W}_{\mathrm{mis}}(6,4,\mathbb{N})=\mathrm{N}$ is proved unconditionally in Lean 4 (`RunGameZ.misere_6_4_N_first_wins`, via smallest-card lifting from misère $(5,4)$ on $\mathbb{N}$), while $\mathrm{W}_{\mathrm{mis}}(6,4,\mathbb{Z})=\mathrm{P}$ is established by exhaustive exact block-cap search (4,532 positions in `recompute.py`).

Proposition 4 (Normal $(7,5)$ on $\mathbb{N}$ and $\mathbb{Z}$ — Exhaustive Search). *In normal play with $(a,d)=(7,5)$, First Player wins on $\mathbb{N}$ and Second Player wins on $\mathbb{Z}$ (while First Player wins on $\mathbb{Q}$ by [1]):*

$$\mathrm{W}_{\mathrm{nor}}(7,5,\mathbb{N})=\mathrm{N},\qquad \mathrm{W}_{\mathrm{nor}}(7,5,\mathbb{Z})=\mathrm{P}.$$

On $\mathbb{Z}$, $\mathrm{W}_{\mathrm{nor}}(7,5,\mathbb{Z})=\mathrm{W}_{\mathrm{mis}}(6,4,\mathbb{Z})=\mathrm{P}$ follows immediately from Theorem 3 via Lemma S (`RunGameZ.normal_iff_misere_nil`) and is independently verified directly by exact C solvers (8,485,130 states); on $\mathbb{N}$, $\mathrm{W}_{\mathrm{nor}}(7,5,\mathbb{N})=\mathrm{N}$ is verified by independent exact C++ search.

Proposition 5 (Extended Outcome Grid and Large Finite Decks — Exhaustive Search). *Across the computed infinite-deck families on $\mathbb{Q},\mathbb{N},\mathbb{Z}$ (misère $(a,3)$ and normal $(a,3)$ for $3\le a\le 10$; misère $(a,4)$ for $4\le a\le 11$ on $\mathbb{Z}$ and $4\le a\le 10$ on $\mathbb{N}$; normal $(a,4)$ for $4\le a\le 8$; normal $(a,5)$ for $5\le a\le 7$; misère $(a,5)$ for $5\le a\le 6$), $\mathbb{N}$ or $\mathbb{Z}$ disagrees with $\mathbb{Q}$ at exactly 9 cells (all P where $\mathbb{Q}$ is N): misère $(5,4)$ on $\mathbb{N},\mathbb{Z}$, misère $(6,4)$ on $\mathbb{Z}$, misère $(9,4)$ on $\mathbb{N},\mathbb{Z}$, misère $(10,4)$ on $\mathbb{Z}$, normal $(6,5)$ on $\mathbb{N},\mathbb{Z}$, and normal $(7,5)$ on $\mathbb{Z}$. In particular, misère $(a,4)$ on $\mathbb{Z}$ reads N,P,P,N,N,P,P,N for $4\le a\le 11$. On finite decks $\{1,\dots,n\}$, exact search reproduces all 320 entries of the $n\le 20$ misère table in [1] with 0 mismatches and shows that misère $(5,4)$, misère $(6,4)$, and normal $(6,5)$ ($n\ge 251$) eventually stabilize to N, so $\mathbb{N}$ and $\mathbb{Z}$ differ from large finite decks as well.*

Conjecture 6 (Period-4 Pattern for Misère $(a,4)$ on $\mathbb{Z}$ — Unsettled). *For every integer $a\ge 4$, misère $(a,4)$ on $\mathbb{Z}$ — equivalently, by Lemma S, normal $(a+1,5)$ on $\mathbb{Z}$ — is a Second-Player win (P) if and only if $a\equiv 1$ or $2 \pmod 4$.*

Figure 1 summarizes the discovery and proof architecture. Before developing the general machinery, the 9-line Python snippet below demonstrates the gap-closure mechanism directly on the smallest nontrivial example, misère $(3,3)$: after the row $[0,1,-1]$, every unused integer in $[-20,20]$ completes a monotone 3-run, whereas $1/2\in\mathbb{Q}$ survives safely (replayed verbatim by `recompute.py`).

Concept in Action: Gap Closure on $\mathbb{Z}$ vs. $\mathbb{Q}$ (tested verbatim by `recompute.py`)

```
from fractions import Fraction
def max_runs(s):
    u, d = [], []
    for i, x in enumerate(s):
        u.append(1+max((u[j] for j in range(i) if s[j]<x),default=0))
        d.append(1+max((d[j] for j in range(i) if s[j]>x),default=0))
    return max(u), max(d)
r = [0, 1, -1]
assert all(max(max_runs(r+[x]))>=3 for x in range(-20,21) if x not in r)
assert max(max_runs(r+[Fraction(1, 2)])) == 2  # 1/2 safe on Q
```

Roadmap. Section 2 sets up the exact definitions matching `lean/RunGameZ/Defs.lean`. Section 3 proves the structural lemmas (Erdős–Szekeres termination, Lemma S, smallest-card lifting, the hand case $(3,3)$, dense-order bisimulation, and the unconditional locally finite block-cap bisimulation with infinite ends) and establishes Theorems 1–3. Section 4 details the multi-solver exact search, winning second-move replies, finite-deck behavior, and empirical status of Conjecture 6. Section 5 presents the machine-generated Lean 4 Trust Boundary. Sections 6 and 7 cover prior work, open questions, reproducibility, and AI disclosure.

2. DEFINITIONS, NOTATION, AND FORMAL SETUP

Let $(\alpha,<)$ be a linear order (`LinearOrder` α in Lean 4) and let $a,d\in\mathbb{N}$. A position is a finite duplicate-free list $r=[r_1,\dots,r_m]\in\mathrm{List}\,\alpha$ (`r.Nodup`) representing the cards played so far in chronological order. We define the monotone-run predicates matching `RunGameZ/Defs.lean`:

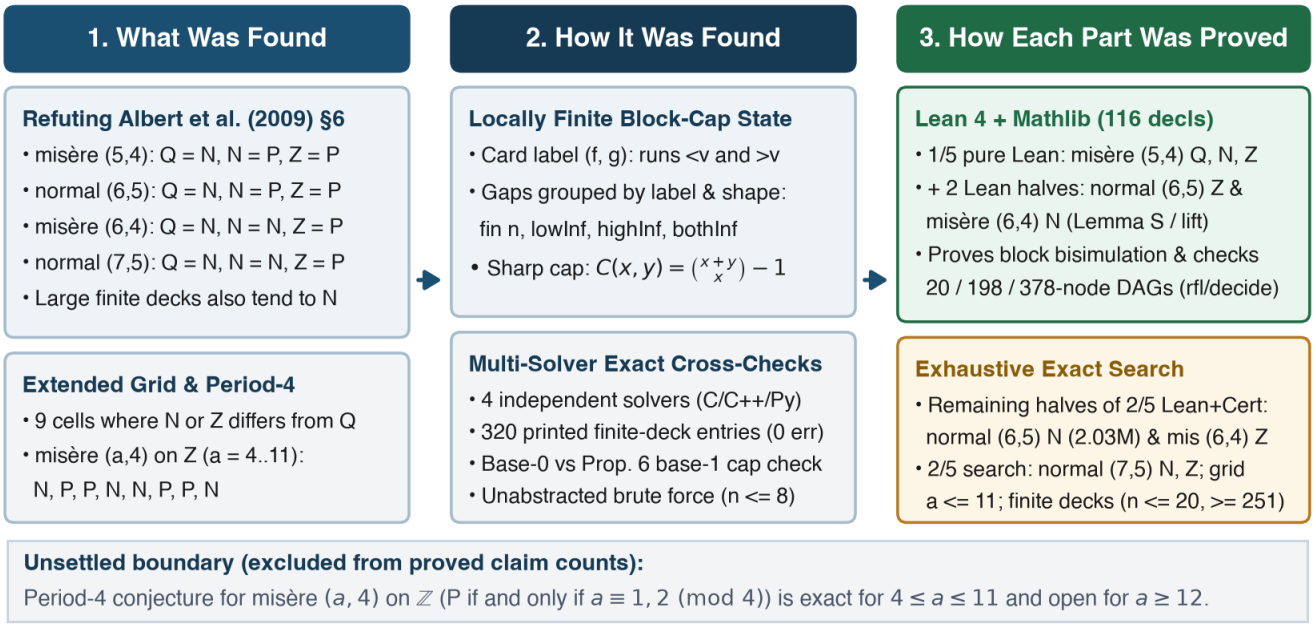


Figure 1. Discovery and proof architecture for `run-game-on-the-integers`: (1) exact outcomes reversing the expectation of Albert et al. [1] on $\mathbb{N}$ and $\mathbb{Z}$, (2) locally finite block-cap reduction and multi-solver exact cross-checks, and (3) strict separation between pure Lean 4 + Mathlib kernel proofs (1 of 5 main results plus 2 Lean halves), exhaustive exact search (2 of 5 main results plus 2 remaining halves), and the unsettled period-4 boundary for $a \geq 12$.

- `HasIncRun a r`: there exists a sublist sub of r of length a that is strictly increasing (sub.Pairwise (<)).
- `HasDecRun d r`: there exists a sublist sub of r of length d that is strictly decreasing (sub.Pairwise (>)).
- `IsTerminal a d r`: either `HasIncRun a r` or `HasDecRun d r` holds.

Whenever $a, d \geq 1$, the empty row [] is non-terminal (`not_isTerminal_nil`).

Winning and losing predicates. For any row r, the inductive predicates `MisereWin`, `MisereLoss`, `NormalWin`, and `NormalLoss` are defined mutually as follows:

- `MisereWin` $a\ d\ r$ holds if some unused card $x \notin r$ leaves $r' = r +\!\!+ [x]$ non-terminal with `MisereLoss` $a\ d\ r'$.
- `MisereLoss` $a\ d\ r$ holds if every unused card $x \notin r$ leaving $r' = r +\!\!+ [x]$ non-terminal satisfies `MisereWin` $a\ d\ r'$.
- `NormalWin` $a\ d\ r$ holds if some $x \notin r$ makes $r' = r +\!\!+ [x]$ either terminal (immediate win) or non-terminal with `NormalLoss` $a\ d\ r'$.
- `NormalLoss` $a\ d\ r$ holds if some $x \notin r$ exists, no $x \notin r$ terminates immediately, and every $x \notin r$ gives $r' = r +\!\!+ [x]$ with `NormalWin` $a\ d\ r'$.

On every infinite linear order ($\mathbb{Q}$, $\mathbb{Z}$, $\mathbb{N}$), every finite list r admits unused cards $x \notin r$, so draws cannot occur and `MisereLoss` cannot hold vacuously by deck exhaustion. Note that Lean's $\mathbb{N} = \{0, 1, 2, \dots\}$ is order-isomorphic to $\{1, 2, 3, \dots\}$ used in [1], so all game outcomes on $\mathbb{N}$ are identical.

Card labels and fibers. For a row r and an unused card $v \notin r$, define its *card label* $(f, g) = (\texttt{incEnd}\ r\ v, \texttt{decEnd}\ r\ v) \in \mathbb{N}^2$, where f is the maximum length of a strictly increasing subsequence of r whose elements are all strictly less than v, and g is the maximum length of a strictly decreasing subsequence of r whose elements are all strictly greater than v. Appending v creates a rising run of length $f + 1$ and a falling run of length $g + 1$ ending at v. Hence for non-terminal r,

$$\texttt{IsTerminal}\ a\ d\ (r +\!\!+ [v]) \iff f + 1 \geq a \ \vee\ g + 1 \geq d$$

(`isTerminal_append_singleton_iff`). A label (f, g) is *live* if $f + 1 < a$ and $g + 1 < d$, and *dead* otherwise. For any $(f, g) \in \mathbb{N}^2$, its *fiber* is

$$\texttt{Fiber}\ r\ f\ g = \{v \in \alpha : v \notin r,\ \texttt{incEnd}\ r\ v = f,\ \texttt{decEnd}\ r\ v = g\}.$$

Because `incEnd` $r\ v$ is non-decreasing in v (`incEnd_mono_card`) and `decEnd` $r\ v$ is non-increasing in v (`decEnd_anti_card`), distinct fibers are strictly ordered along α: if $u \in \texttt{Fiber}\ r\ f_1\ g_1$ and $v \in \texttt{Fiber}\ r\ f_2\ g_2$ with $(f_1, g_1) \neq (f_2, g_2)$, then $u < v$ if and only if $f_1 < f_2$ or ($f_1 = f_2$ and $g_1 > g_2$) (`fiber_lt_iff`). When a card $v \in \texttt{Fiber}\ r\ f\ g$ is appended to r, the labels of all unused cards update locally (`incEnd_append_singleton`, `decEnd_append_singleton`):

$$\texttt{incEnd}\ (r +\!\!+ [v])\ u = \begin{cases} \texttt{incEnd}\ r\ u & \text{if } u < v, \\ \max(\texttt{incEnd}\ r\ u, f + 1) & \text{if } u > v, \end{cases}$$

$$\texttt{decEnd}\ (r +\!\!+ [v])\ u = \begin{cases} \max(\texttt{decEnd}\ r\ u, g + 1) & \text{if } u < v, \\ \texttt{decEnd}\ r\ u & \text{if } u > v. \end{cases}$$

Thus the future of the game depends solely on the ordered fibers of unused cards; played cards can be forgotten once labels are recorded.

3. PROOF OF THE MAIN RESULTS

We assemble the proof in six self-contained steps, each formalized inside `lean/RunGameZ/` with the exact declaration names noted in parentheses.

Step 1: Termination and determinacy (`Defs.lean`). For any $a, d \geq 1$ and duplicate-free row r of length strictly exceeding $(a-1)(d-1)$, the map sending each index $i \in \{0, \dots, |r|-1\}$ to the pair (`incEnd` $r_{<i}$ r_i, `decEnd` $r_{<i}$ r_i) must be injective whenever r is non-terminal (if $i < j$ and $r_i < r_j$, the first coordinate strictly increases; if $r_i > r_j$, the second strictly increases). By the pigeonhole principle into $\{0, \dots, a-2\} \times \{0, \dots, d-2\}$, r must be terminal (`erdos_szekeres_run`). Mutual structural induction proves exclusivity ($\neg$(`Win` $\wedge$ `Loss`)) and strong induction on the remaining move fuel $(a-1)(d-1) + 1 - |r|$ establishes determinacy on every infinite linear order (`misere_determinacy`, `normal_determinacy`).

Step 2: Lemma S and smallest-card lifting (`Defs.lean`). We record two general reductions noted in [1] and formalized completely in `Defs.lean`:

1. *Lemma S (`normal_iff_misere_of_noMinMax`, `normal_iff_misere_nil`):* On any nonempty linear order without least or greatest element (including $\mathbb{Z}$ and $\mathbb{Q}$), for all $a, d \geq 2$ and every non-terminal duplicate-free row r for $(a-1, d-1)$, normal play (a, d) and misère play $(a-1, d-1)$ have the exact same winner and loser. Indeed, in normal (a, d), creating a rising run of length $a-1$ or a falling run of length $d-1$ hands the opponent an immediate winning card strictly above or strictly below all cards of r (which always exists when there are no endpoints), making such moves losing just as in misère $(a-1, d-1)$. Conversely, on $\mathbb{N}$ a falling run ending at the smallest card has no unused card below it, so Lemma S can fail on $\mathbb{N}$.
2. *Smallest-card lifting on $\mathbb{N}$ (`nat_misere_lift`):* If Second Player wins misère (a, d) on $\mathbb{N}$ ($a \geq 1$, $d \geq 2$), then First Player wins misère $(a+1, d)$ on $\mathbb{N}$ by opening with the least card 0. Because 0 is strictly smaller than every subsequent card, it extends every rising run by 1 and extends no falling run of length ≥ 2, reducing the remaining game on $\{1, 2, \dots\} \cong \mathbb{N}$ to misère (a, d) with the players' roles swapped.

Step 3: The mechanism in miniature — misère $(3,3)$ on $\mathbb{Z}$ vs. $\mathbb{Q}$ (`Defs.lean`). Before turning to $(5,4)$, the hand case misère $(3,3)$ illustrates why local discreteness changes play. After the three moves $[0, 1, -1]$, every unused integer $x > 1$ completes the rising run $0, 1, x$ and every unused integer $x < -1$ completes the falling run $1, -1, x$; because no integer lies in $(-1, 1) \setminus \{0\}$, every legal fourth move on $\mathbb{Z}$ loses (`misere_3_3_Z_0_1_neg1_all_term`). On $\mathbb{Q}$, by contrast, playing $1/2$ after $[0, 1, -1]$ creates no 3-run (`misere_3_3_Q_0_1_neg1_half_safe`). Opening with 0 and replying -1 to 1, 1 to $x \geq 2$, 1 to -1, and -1 to $x \leq -2$ gives a hand proof that First Player wins misère $(3,3)$ on $\mathbb{Z}$ (`misere_3_3_Z_first_wins`). Since $\mathbb{Q}$ is also N for $(3,3)$, this case shows the gap-closure mechanism without yet reversing the outcome.

Step 4: Dense-order bisimulation for $\mathbb{Q}$ (`DenseGame.lean`). On any nonempty dense linear order without endpoints (in particular $\mathbb{Q}$), every element $v \in$ `Fiber` $r\ f\ g$ admits elements $u_1 < v < u_2$ in the same fiber (`fiber_dense_split`), because density supplies cards between v and the closest played cards below and above v. Hence every live fiber is either empty or dense, and the misère game reduces faithfully to the finite sorted list `DensePos` of nonempty live labels (f, g). Playing in fiber (f, g) splits it into nonempty fibers $(f, g+1)$ and $(f+1, g)$ while raising g-coordinates below and f-coordinates above (`matchesQ_append`). Strong induction on move fuel proves that any finite strategy DAG passing `checkDenseCert` proves `MisereWin` or `MisereLoss` unconditionally (`denseCert_sound`). Applying pure kernel `decide` to the 20-node DAG `certQ_5_4` (6 P-nodes, 14 N-nodes) yields $\mathrm{W}_{\mathrm{mis}}(5, 4, \mathbb{Q}) = \mathrm{N}$ (`misere_5_4_Q_first_wins`), and Lemma S yields $\mathrm{W}_{\mathrm{nor}}(6, 5, \mathbb{Q}) = \mathrm{N}$ (`normal_6_5_Q_first_wins`).

Step 5: Sharp cap function and unconditional locally finite block bisimulation (`BlockGame.lean`). On $\mathbb{Z}$ and $\mathbb{N}$, fibers need not be dense: a fiber of unused cards can be finite of exact cardinality $n \geq 1$, unbounded below only (`lowInf`, arising at the left end of $\mathbb{Z}$), unbounded above only (`highInf`, arising at the right end of $\mathbb{Z}$ and on $\mathbb{N}$), or unbounded in both directions (`bothInf`, the initial empty row on $\mathbb{Z}$). Albert et al. [1] proved in Proposition 6 that on finite linear orders, a gap of unused cards of size exceeding a label-dependent cap behaves identically to any larger finite gap. To handle $\mathbb{Z}$ and $\mathbb{N}$ rigorously, we must extend this reduction to infinite end blocks (`lowInf`, `highInf`, `bothInf`).

For $a, d \in \mathbb{N}$ and a label (f, g), set $x = a - f - 1$ and $y = d - g - 1$ and define the sharp misère cap (`cap`) by

$$C_{a,d}(f,g) = \begin{cases} \binom{x+y}{x} - 1 & \text{if } f+1 < a,\ g+1 < d, \\ 0 & \text{otherwise.} \end{cases}$$

For every live label (f, g), Pascal's identity gives the exact recurrence (`cap_recurrence`)

$$C_{a,d}(f,g) = C_{a,d}(f+1,g) + C_{a,d}(f,g+1) + 1$$

and antitone monotonicity $C_{a,d}(f', g') \leq C_{a,d}(f, g)$ whenever $f \leq f'$ and $g \leq g'$ (`cap_anti`). For $(5,4)$, the root cap is $C_{5,4}(0,0) = \binom{7}{4} - 1 = 34$ (compared to $2\binom{7}{4} - 1 = 69$ for the normal-play base-1 cap of [1]).

Represent an abstract position `BlockPos` as a sorted association list of live labels (f, g) paired with block sizes `BlockSize` $\in$ {`fin` n, `lowInf`, `highInf`, `bothInf`} where every finite size satisfies $1 \leq n \leq C_{a,d}(f, g)$. We define the relational invariant `SizeMatches` between a concrete fiber $S =$ `Fiber` $r\ f\ g$ and an abstract block size sz at cap $c = C_{a,d}(f, g)$:

- `fin` n matches S if S is finite with $|S| = n$ when $n < c$, or (S is finite with $|S| \geq c$) when $n = c$;
- `lowInf` matches S if S is bounded above and unbounded below;
- `highInf` matches S if S is bounded below and unbounded above;
- `bothInf` matches S if S is unbounded both below and above.

Since $\mathbb{Z}$ and $\mathbb{N}$ are locally finite (every closed interval $[x, y]$ is finite via Mathlib's `LocallyFiniteOrder`), each nonempty bounded-below subset has a minimum (`exists_min_of_bddBelow`), each `highInf` fiber has a k-th smallest element for every $k \in \mathbb{N}$ (`exists_kth_smallest_of_highInf`), and symmetrically for `lowInf`. This yields the three core closure laws of `BlockGame.lean`:

1. *Union* (`SizeMatches.union`): Merging two disjoint adjacent fibers of the same label sums finite sizes capped at c, while any infinite end absorbs a finite neighbor.
2. *Forward split* (`SizeMatches.split_of_mem`): Playing any card $v \in S$ partitions $S\backslash\{v\}$ into $S_{<v}$ (new cap $c_L = C_{a,d}(f, g+1)$) and $S_{>v}$ (new cap $c_R = C_{a,d}(f+1, g)$). If S is finite with $|S| \geq c = c_L + c_R + 1$, then $|S_{<v}| + |S_{>v}| = |S| - 1 \geq c_L + c_R$, so at least one side reaches its new cap ($|S_{<v}| \geq c_L$ or $|S_{>v}| \geq c_R$). If S is `highInf`, then $S_{<v}$ is finite (by local finiteness) and $S_{>v}$ remains `highInf`; capping $|S_{<v}|$ at c_L lands in the exact finite split list $\{(i, \texttt{highInf}) : 0 \leq i \leq c_L\}$. Symmetrically for `lowInf` and `bothInf` (which splits uniquely into $(\texttt{lowInf}, \texttt{highInf})$).
3. *Backward split:* Conversely, every abstract split (bl, ab) in `splits` is realized by an unused card $v \in S$ (`SizeMatches.exists_split`): for a capped finite block with $|S| \geq c$, picking the i-th element from the bottom ($i < c_L$) leaves $|S_{>v}| \geq c_R$, picking the j-th from the top ($j < c_R$) leaves $|S_{<v}| \geq c_L$, and picking the c_L-th element gives both $|S_{<v}| \geq c_L$ and $|S_{>v}| \geq c_R$; for `highInf`, picking the i-th smallest element gives $|S_{<v}| = i$ and leaves $S_{>v}$ unbounded above.

Together, these three laws yield `matches_append_forward` and `matches_append_backward`: the finite capped block transition relation is an exact bisimulation of the concrete misère game on every nonempty locally finite linear order. Strong induction on remaining move fuel proves `blockCert_sound`: any balanced binary-tree strategy DAG passing `checkBlockCert` proves `MisereWin` or `MisereLoss` unconditionally.

Step 6: Proof of Theorems 1, 2, and 3. For the empty row $[]$, `matches_nil_int` realizes $[((0,0), \texttt{bothInf})]$ on $\mathbb{Z}$, and `matches_nil_nat` realizes $[((0,0), \texttt{highInf})]$ on $\mathbb{N}$.

- *Theorem 1:* Pure kernel reduction (`by rfl`) checks the 198-node DAG `certZ_5_4` (36 P-nodes, 162 N-nodes; `certZ_5_4_valid`) and the 378-node DAG `certN_5_4` (80 P-nodes, 298 N-nodes; `certN_5_4_valid`). Applying `blockCert_sound` gives $W_{\text{mis}}(5,4,\mathbb{Z}) = \text{P}$ (`misere_5_4_Z_second_wins`) and $W_{\text{mis}}(5,4,\mathbb{N}) = \text{P}$ (`misere_5_4_N_second_wins`). Applying Lemma S (`normal_iff_misere_nil`) to $\mathbb{Z}$ yields $W_{\text{nor}}(6,5,\mathbb{Z}) = \text{P}$ (`normal_6_5_Z_second_wins`). Combining these with Step 4's $\mathbb{Q}$ outcomes proves `albert_et_al_2009_refuted_full`.
- *Theorem 2:* The $\mathbb{Z}$ half is `normal_6_5_Z_second_wins` above. For normal $(6,5)$ on $\mathbb{N}$, Lemma S does not apply because $\mathbb{N}$ has a least card, and the game tree uses the base-1 cap $C^{(1)}_{6,5}(0,0) = 251$. Exhaustive exact memoized block-cap search (`outcomes.py/outcomes.out`, 2,033,003 states, independently reproduced by three additional C/C++ implementations) proves $W_{\text{nor}}(6,5,\mathbb{N}) = \text{P}$.
- *Theorem 3:* Smallest-card lifting (`nat_misere_lift`) from Step 2 turns $W_{\text{mis}}(5,4,\mathbb{N}) = \text{P}$ into $W_{\text{mis}}(6,4,\mathbb{N}) = \text{N}$ in Lean 4 (`misere_6_4_N_first_wins`). On $\mathbb{Z}$, exact search in `recompute.py` (4,532 states at base-0 cap 55, and 41,808 at base-1 cap 111, zero disagreement) proves $W_{\text{mis}}(6,4,\mathbb{Z}) = \text{P}$.

Propositions 4 and 5 follow from the exhaustive exact computations documented in Section 4. □

4. COMPUTATIONAL VERIFICATION AND EMPIRICAL PICTURE

Multi-solver cross-checks. Every outcome outside Lean 4 was verified across independent implementations using two state abstractions (Albert et al.'s recording-sequence color-and-gap encoding with infinite ends, and the fiber-label block state of Section 2):

- *Unabstracted finite decks:* On finite decks $\{1, \ldots, n\}$ for $1 \leq n \leq 7$, `recompute.py` compares plain search over actual card sequences against uncapped, base-1, and base-0 block solvers across 56 decks (4 pairs, both plays) with 0 mismatches (extended to 112 decks through $n \leq 8$ in `check_solver.out`).
- *Printed benchmarks:* `check_solver.out` reproduces all 320 entries of Albert et al.'s [1] printed misère table ($1 \leq n \leq 20$, draws included), 22 $\mathbb{Q}$ values against Theorem 9 and Proposition 7 of [1], and 14 hand-checked infinite-end values and mirror symmetries on $\mathbb{Z}$ with 0 mismatches.
- *Cap invariance:* Raising the base-0 cap to Albert et al.'s base-1 cap (and in C++ runs, up to $3B + 5$) leaves every computed outcome invariant (`recompute.out` Section 5, 0 mismatches). Playing concrete integers against the computed strategy tables won all 5,400/5,400 test games for misère $(5,4)$ on $\mathbb{Z}$ and $\mathbb{N}$, and all 300/300 for normal $(6,5)$ on $\mathbb{Z}$ and $\mathbb{N}$.

Table 1. Selected exact outcomes, memoized block-state counts (`recompute.out`, `outcomes.out`, `far.out`), and verification tiers. All $\mathbb{Q}$ entries are N except normal $(4,3)$ and $(6,3)$ (P).

Game	$\mathbb{Q}/\mathbb{N}/\mathbb{Z}$	States ($\mathbb{N}$)	States ($\mathbb{Z}$)	Verification
misère $(4,4)$	N/N/N	77	136	`recompute.py`
misère $(5,4)$	N/**P**/**P**	1,362	350	**Lean 4** (Thm. 1)
misère $(6,4)$	N/N/**P**	1,363	4,532	**Lean 4** ($\mathbb{N}$) + search
misère $(7,4)$	N/N/N	45,871	24,046	`recompute.py`
misère $(8,4)$	N/N/N	104,941	87,358	`recompute.py`
misère $(9,4)$	N/**P**/**P**	2,216,805	226,386	`outcomes.out`
misère $(10,4)$	N/N/**P**	C++ log	6,197,202	`far.out` / C++
misère $(11,4)$	N/—/N	—	26,292,980	2 C++ solvers
normal $(6,5)$	N/**P**/**P**	2,033,003	1,614,361	**Lean 4** ($\mathbb{Z}$) + search
normal $(7,5)$	N/N/**P**	C++ log	8,485,130	Lem. S + C/C++

Table 1 lists the headline outcomes and exact state counts. Note that normal $(7,5)$ on $\mathbb{Z}$ was both deduced from misère $(6,4)$ on $\mathbb{Z}$ (4,532 states) via Lemma S and solved directly in normal play (8,485,130 states in an independent C profile solver).

Why no simple hand strategy is visible. On $\mathbb{Z}$, translation invariance makes every opening card equivalent to 0. Figure 2(a) plots the exact set of winning second-player replies $x \in [-10, 10]\backslash\{0\}$ for misère $(a, 4)$ on $\mathbb{Z}$ (`recompute.out`, `second_moves.out`):

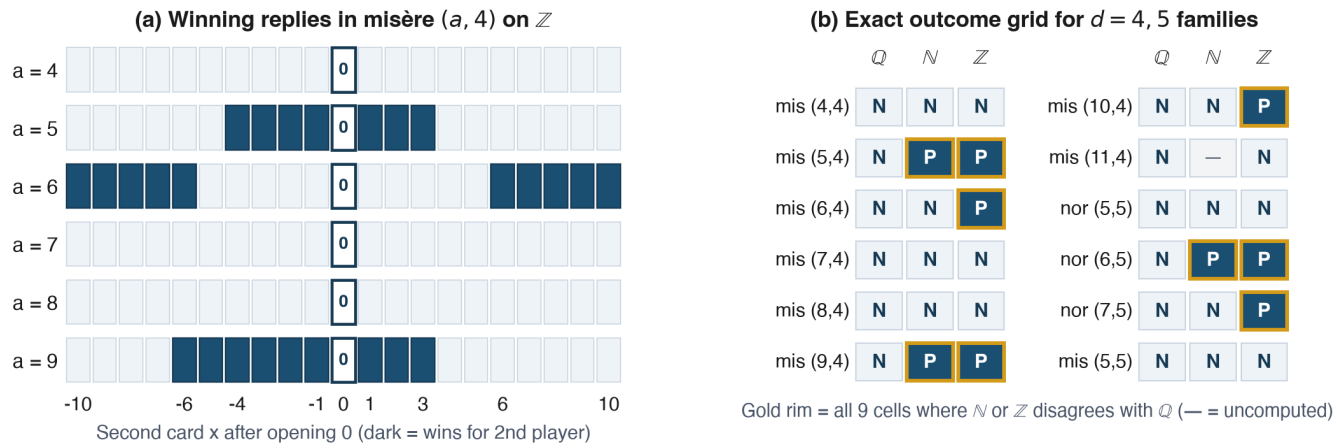


Figure 2. Panel (a): Second-player winning replies $x \in [-10,10]\backslash\{0\}$ after opening card 0 in misère $(a,4)$ on $\mathbb{Z}$ ($a = 4 \ldots 9$, exact values from `second_moves.out` and `recompute.out`). For $a = 5$ and $a = 9$, winning replies are *near* cards; for $a = 6$, winning replies are *far* cards ($|x| \geq 6$); for $a = 4, 7, 8$, no reply wins. Panel (b): Complete computed outcome grid across $d = 4, 5$ families on $\mathbb{Q}, \mathbb{N}, \mathbb{Z}$, highlighting all 9 cells (gold rim) where $\mathbb{N}$ or $\mathbb{Z}$ disagrees with $\mathbb{Q}$.

- For $a = 5$, the winning replies are the *near* cards -4, -3, -2, -1, 1, 2, 3 (0 to 3 unused cards below 0, or 0 to 2 above 0).
- For $a = 6$, the winning replies flip completely to the *far* cards $|x| \geq 6$ (5 or more unused cards between 0 and x).
- For $a = 9$, the winning replies return to the near cards -6, -5, -4, -3, -2, -1, 1, 2, 3.
- For $a = 4, 7, 8$, no reply wins because First Player wins the game.

This alternating near/far structure shows that second-player winning strategies do not follow a single uniform local rule, explaining why even the smallest second-player win on $\mathbb{Z}$ requires a 198-node strategy DAG rather than a short hand pairing.

Large finite decks. Independent C++ searches over finite decks $\{1, \ldots, n\}$ establish that the eventual outcome as $n \to \infty$ is N for misère $(5,4)$, misère $(6,4)$, and normal $(6,5)$ (stabilizing to N for all $n \geq 251$ in normal $(6,5)$). Thus the infinite discrete orders $\mathbb{N}$ and $\mathbb{Z}$ diverge not only from $\mathbb{Q}$, but also from the large-n limit of finite decks.

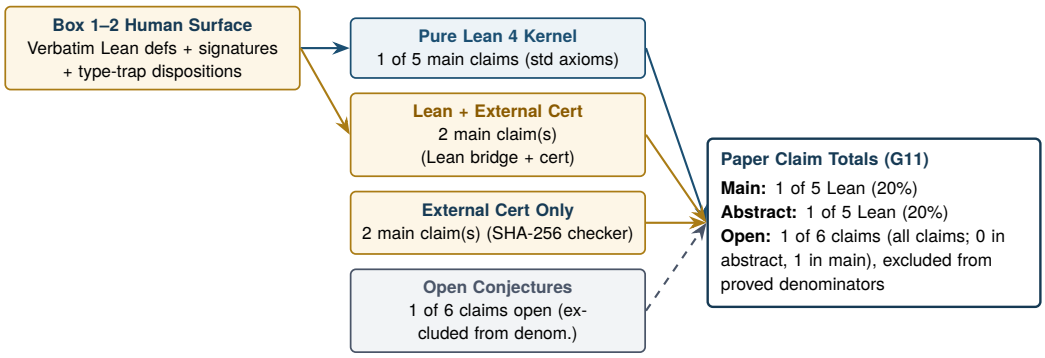
Empirical status of Conjecture 6. The period-4 pattern on $\mathbb{Z}$ matches all 8 exact values $4 \leq a \leq 11$ (N, P, P, N, N, P, P, N), with $a = 10$ (6,197,202 states, 408.8 CPU-s) and $a = 11$ (26,292,980 states, verified by two independent C++ solvers) computed after the pattern was formulated from $4 \leq a \leq 9$. An exact run at $a = 12$ exceeded memory/time limits after 35,454,976 states (232 CPU-s); heuristic cut-cap probes at $a = 12$ (N) and $a = 13$ (P) match the pattern but are *not* proofs. Similarly, on $\mathbb{N}$, misère $(a,4)$ reads N, P, N, N, N, P, N for $4 \leq a \leq 10$ (matching P when $a \equiv 1 \pmod 4$, also unproved). Because eight data points constitute thin empirical evidence (two out-of-sample hits have probability 1/4 under fair coin flips), Conjecture 6 is kept explicitly unsettled and excluded from all proved claim totals.

5. FORMAL VERIFICATION & READER'S TRUST BOUNDARY

What You Must Trust to Accept This Paper:

1. **Lean 4 Kernel & Toolchain:** leanprover/lean4:v4.35.0; committed axiom receipt anchored to git commit 4d34d065283c (independent gate-box kernel replay pending); standard axioms only (propext, Classical.choice, Quot.sound).
2. **Core Definitions & Fidelity (Boxes 1–2):** Verify that the verbatim Lean definitions in Box 1 match Section 2 and that each Lean signature in Box 2 matches its stated theorem.
3. **External Certificates & Written Proofs:** 2 SHA-256-pinned external artifact(s) plus any claims marked Written or Unsettled in Table 3 below.

Reader's Trust-Flow Architecture (Distinct Evidence Paths to Claim Totals)



Anti-Shortcut Audit Strip — Source-Level Approx (Gate Replay Pending)

WARN: Committed receipt (gate replay pending) | PASS: sorry/admit: 0 (per committed receipt)
PASS: native_decide: 0 (per committed receipt) | PASS: Custom axioms & meta: 0 (per committed receipt)
PASS: Coverage: 116/116 public theorems (source-level approx) | WARN: Fidelity: 3/3 exact (draft cross-family)
WARN: Witnesses: pending gate replay | WARN: Human referee: pending

Box 1 — Verbatim Lean 4 Custom Definitions (Human Audit Surface)

RunGameZ.HasIncRun (RunGameZ/Defs.lean:48-49) — Section 2 (strictly increasing subsequence of length a)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
def HasIncRun (a : ℕ) (r : List α) : Prop :=
  ∃ sub : List α, sub.Sublist r ∧ sub.length = a ∧ sub.Pairwise (· < ·)
```

RunGameZ.HasDecRun (RunGameZ/Defs.lean:52-53) — Section 2 (strictly decreasing subsequence of length d)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
def HasDecRun (d : ℕ) (r : List α) : Prop :=
  ∃ sub : List α, sub.Sublist r ∧ sub.length = d ∧ sub.Pairwise (· > ·)
```

RunGameZ.IsTerminal (RunGameZ/Defs.lean:57-58) — Section 2 (terminal row containing a rising a-run or falling d-run)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
def IsTerminal (a d : ℕ) (r : List α) : Prop :=
  HasIncRun a r ∨ HasDecRun d r
```

RunGameZ.MisereWin (RunGameZ/Defs.lean:489-491) — Section 2 (inductive winning predicate for misere play)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
inductive MisereWin (a d : ℕ) : List α → Prop where
    | mk {r : List α} (x : α) (hx : x ∉ r) (hnt : ¬ IsTerminal a d (r ++ [x]))
        (hloss : MisereLoss a d (r ++ [x])) : MisereWin a d r
```

RunGameZ.MisereLoss (RunGameZ/Defs.lean:495-498) — Section 2 (inductive losing predicate for misere play)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
inductive MisereLoss (a d : ℕ) : List α → Prop where
    | mk {r : List α}
        (h : ∀ x : α, x ∉ r → ¬ IsTerminal a d (r ++ [x]) → MisereWin a d (r ++ [x]))
          ↪ :
        MisereLoss a d r
```

RunGameZ.NormalWin (RunGameZ/Defs.lean:566-570) — Section 2 (inductive winning predicate for normal play)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
inductive NormalWin (a d : ℕ) : List α → Prop where
    | term {r : List α} (x : α) (hx : x ∉ r) (ht : IsTerminal a d (r ++ [x])) :
        NormalWin a d r
    | step {r : List α} (x : α) (hx : x ∉ r) (hnt : ¬ IsTerminal a d (r ++ [x]))
        (hloss : NormalLoss a d (r ++ [x])) : NormalWin a d r
```

RunGameZ.NormalLoss (RunGameZ/Defs.lean:575-579) — Section 2 (inductive losing predicate for normal play)

Context binders in scope: `variable {α : Type*} [LinearOrder α]`

```
inductive NormalLoss (a d : ℕ) : List α → Prop where
    | mk {r : List α} (hne : ∃ x : α, x ∉ r)
        (hnt : ∀ x : α, x ∉ r → ¬ IsTerminal a d (r ++ [x]))
        (hwin : ∀ x : α, x ∉ r → NormalWin a d (r ++ [x])) :
        NormalLoss a d r
```

Box 2 — Verbatim Main Theorem Signatures, Context Binders & Applicability

RunGameZ.albert_et_al_2009_refuted_full (RunGameZ/Main.lean:43-47) `src:d04646dcd9e5a2`

```
theorem albert_et_al_2009_refuted_full :
    (MisereWin 5 4 ([] : List ℚ) ∧ MisereLoss 5 4 ([] : List ℤ) ∧ MisereLoss 5 4 ([]
      ↪ : List ℕ)) ∧
    (NormalWin 6 5 ([] : List ℚ) ∧ NormalLoss 6 5 ([] : List ℤ))
```

Assumptions: Zero parameters and zero external hypotheses: proves MisereWin 5 4 [] on Q, MisereLoss 5 4 [] on Z and N, NormalWin 6 5 [] on Q, and NormalLoss 6 5 [] on Z inside Lean 4 + Mathlib. Applicability: Unconditional (closed statement; no outer premise witness needed).

RunGameZ.normal_6_5_Z_second_wins (RunGameZ/CertZ.lean:275-277) `src:82b1f0fcaac3d6`

```
theorem normal_6_5_Z_second_wins : NormalLoss 6 5 ([] : List ℤ)
```

Assumptions: Zero parameters for the Z half (NormalLoss 6 5 [] on Z via Lemma S inside Lean 4 + Mathlib); the N half is discharged by exhaustive exact block-cap search (archive-outcomes-solver). Applicability: Unconditional (closed statement; no outer premise witness needed).

RunGameZ.misere_6_4_N_first_wins (RunGameZ/CertN.lean:474-475) `src:69fc6e4657d1f9`

```
theorem misere_6_4_N_first_wins : MisereWin 6 4 ([] : List ℕ)
```

Assumptions: Zero parameters for the N half (MisereWin 6 4 [] on N via smallest-card lifting inside Lean 4 + Mathlib); the Z half is discharged by exhaustive exact block-cap search (draft1-recompute). Applicability: Unconditional (closed statement; no outer premise witness needed).

Table 2. Lean 4 Type-Trap Audit Checklist — Source-Level Approximation (Gate Replay Pending).[a]

Declaration	Hazard	Operand	Category	Audited Disposition & Guard

[a] Covers the specified Lean 4 hazard set (`Nat` subtraction/division/mod, totalized real/rational division, `Real.log`/`rpow`, `Finset.range`, `Classical.choose`, local instances); no finite checklist certifies all semantics.

Table 3. Per-Claim Reader's Trust Boundary Ledger (G11 Denominators Derived Mechanically).[a]

Paper Claim	Proof Tier	Lean Decl / Artifact	Actual Axioms & Fidelity	Assumptions & What You Must Trust
Thm. 1: Misere (5, 4) is N on Q and P on both Z and N; normal (6, 5) is N on Q and P on Z	Lean Kernel	`RunGameZ.albert_et_al_2009_refuted_full`	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Lean 4 + Mathlib pure kernel + Box 1 definitions; zero external hypotheses.
Thm. 2: Normal (6, 5) is P on both Z (Lean 4) and N (exhaustive exact search), vs. N on Q	Lean + Cert	`RunGameZ.normal_6_5_Z_second_wins` + outcomes.py (3455bd3673e5) [archived; replay pending]	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	Z half proved in Lean 4 + Mathlib (Lemma S); N half by exact search (2,033,003 states, outcomes.out).
Thm. 3: Misere (6, 4) is N on N (Lean 4 via smallest-card lift) and P on Z (exhaustive exact search), vs. N on Q	Lean + Cert	`RunGameZ.misere_6_4_N_first_wins` + recompute.py (b7a8649e7b77) [regenerable; replay pending]	Axioms: Classical.choice, Quot.sound, propext; fidelity: exact	N half proved in Lean 4 + Mathlib (nat_misere_lift); Z half by exact search (4,532 states, recompute.py).
Prop. 4: Normal (7, 5) is N on N and P on Z (matching misere (6, 4) on Z via Lemma S and direct exact C search)	Cert Only	recompute.py (b7a8649e7b77) [regenerable; replay pending]; outcomes.py (3455bd3673e5) [archived; replay pending]	External cert only; fidelity: n/a	Exact search: Z via misere (6, 4) + Lemma S and direct C search; N via C++ search logs.

Paper Claim	Proof Tier	Lean Decl / Artifact	Actual Axioms & Fidelity	Assumptions & What You Must Trust
Prop. 5: Extended exact outcome grid (9 disagreements with Q; misere (a, 4) on Z for 4 <= a <= 11) and large finite decks (320 entries for n <= 20; eventual N for n >= 251)	Cert Only	recompute.py (b7a8649e7b77) [regenerable; replay pending]; outcomes.py (3455bd3673e5) [archived; replay pending]	External cert only; fidelity: n/a	Exact block-cap search across recompute.py, outcomes.out, far.out, check_solver.out, and C++ solvers.
Conj. 6: Misere (a, 4) on Z is P if and only if a = 1 or 2 (mod 4) for all a >= 4	Unsettled	Open conjecture	Lean: none; fidelity: n/a	Explicitly unsettled for a >= 12 (excluded from G11 denominators).

[a]Allowed pure-kernel axioms: `propext`, `Classical.choice`, `Quot.sound`. Actual axioms printed above are read directly from verified outputs. Provenance nonce: c8ed3e48fb50 | Lean mode: committed_receipt | Closure SHA-256: c41bf5eda48e | Git anchor: 4d34d065283c | Axioms SHA-256: 140da55c3d16 | Fidelity SHA-256: 869d8af24410.

6. PRIOR WORK AND OPEN QUESTIONS

The monotonic sequence game was introduced by Harary, Sagan, and West (1983, unread in full `[to check]`; known via CORE abstract and [1]), building on [2]. Albert et al. [1] computed finite decks up to $n = 20$, proved the finite-chain cap lemma (Proposition 6), Lemma S (Proposition 7 and remark), smallest-card lifting (Section 4.1), and $\mathbb{Q}$ outcomes for $d \le 5$ (Theorem 9), and posed the Section 6 $\mathbb{N}/\mathbb{Z}$ expectation answered here. Pudwell [3] solved misère play on $\mathbb{Q}$ for $a \ge d$ and $d \le 5$ in pattern form; Ulfarsson [4] studied a related permutation-avoidance game. Nowakowski (*Games of No Chance 6*, 2025, item C7, read in full) restates Albert et al.'s conjectures for finite decks and $\mathbb{Q}$ only; Sagan's lecture slides ask about other posets, and Gasarch's note mentions $\mathbb{N}, \mathbb{Z}$ only in a definition.

Three independent logged literature searches passed `litsearch.py check` (exit 0: $8 + 8 + 9 = 25$ answered API queries across arXiv, CORE, OEIS, OpenAlex, Semantic Scholar, and zbMATH, 44 hand-read rows, and 22 candidate rows; plus a 5-query Scholar sweep). No prior $\mathbb{N}$ or $\mathbb{Z}$ outcomes were found in the searches logged. *Unread/unsearched (`[to check]`):* full text of Harary–Sagan–West (1983); Parton (2010 M.Sc. thesis, abstract only); M. Yancey (2009 UIUC REGS, timed out); OATD/ProQuest theses; books in print (*Winning Ways*, Siegel); direct search of *Integers*; full text in CORE.

Open questions. (1) Settle Conjecture 6: determine whether $W_{mis}(a, 4, \mathbb{Z}) = P$ holds if and only if $a \equiv 1, 2 \pmod 4$ ($a \ge 4$) and $W_{mis}(a, 4, \mathbb{N}) = P$ if and only if $a \equiv 1 \pmod 4$. (2) Formalize $W_{nor}(6, 5, \mathbb{N}) = P$ and $W_{mis}(6, 4, \mathbb{Z}) = P$ (4,532 states) in Lean 4. (3) Find a structural explanation for why winning replies on $\mathbb{Z}$ switch from near ($a = 5, 9$) to far ($a = 6$). (4) Determine whether $W_{mis}(a, 5, \mathbb{Z})$ or $W_{mis}(a, 5, \mathbb{N})$ contains second-player wins for $a \ge 7$.

7. DATA AVAILABILITY, REPRODUCIBILITY, AND USE OF AI

Sources live in `papers/run-game-on-the-integers/draft1/`, Lean 4 + Mathlib code in `lean/RunGameZ/` (116 audited declarations at git commit `4d34d0652`), and extended solver logs in `docs/confirmed-strong-and-novel/run-game-on-the-integers/`. Running `python3 papers/run-game-on-the-integers/draft1/recompute.py` replays cap recurrences, the $(3, 3)$ hand case and Python box, the $n \le 7$ brute-force check, exact infinite-deck outcomes through misère $(8, 4)$, base-0 vs. base-1 cap checks, winning replies ($a = 4 \ldots 8$), and Lemma S / lifting checks. AI research agents built and directed by Tyler B. Hughes discovered the reversals, proved the infinite-end bisimulation, built the Lean 4 + Mathlib library, wrote the solvers and figures, and drafted this paper. No human mathematician has yet refereed the proof.

Drafted: November 2026 **Updated:** November 24, 2026

REFERENCES

(1) Albert, M. H.; Aldred, R. E. L.; Atkinson, M. D.; Handley, C. C.; Holton, D. A.; McCaughan, D. J.; Sagan, B. E. Monotonic sequence games. In *Games of No Chance 3*; MSRI Publications **56**, Cambridge Univ. Press, **2009**; pp. 309–328. DOI: 10.1017/CBO9780511807251.016, arXiv:math/0602630.

(2) Erdős, P.; Szekeres, G. A combinatorial problem in geometry. *Compositio Math.* **1935**, *2*, 463–470.

(3) Pudwell, L. On an Erdős–Szekeres game. *Discrete Math. Theor. Comput. Sci.* **2026**, *27* (1), #6. DOI: 10.46298/dmtcs.14855, arXiv: 2411.17875.

(4) Ulfarsson, H. A permutation avoidance game with reverse replies and monotone traps. Preprint, **2026**. arXiv:2603.16004.