Repetition under a Deadline: Memory Bounds and Capacity Profiles

Mantis Discovery

September 2026

Abstract

We study alternating graph games with optional repetition claims, expiring budgets, and no reset on a directed cycle. A supplied history reduces to a fixed draw boundary in the current strongly connected component and a bit for a current claim. The clock game has positional policies on the arena that are optimal at every entering reserve.

For $N \geq 8$ vertices, capacity H, and repetition threshold k, put $S = (H - 2k + 2)_+$. Exact online value monitoring needs no history memory when $H \leq 2k - 2$. Otherwise it requires

$$\Theta\left(\min\left\{N \log k,\ \log \binom{N+S}{S}\right\}\right)$$

bits, except when $k = 2$ and $H \in \{3, 4\}$, which require exactly one bit. The upper bound discards counts that cannot reach the threshold before expiration; common continuations distinguish equal-length live histories for the lower bound. Randomization with fixed error below $1/2$ does not improve the dense $\Theta(N \log k)$ bound.

For clock games with positive integer costs and arbitrary reset cycles, all capacity profiles together have at most N breakpoint capacities and $N(N + 1)/2$ finite constant runs. The quadratic order is sharp. At most $N + 1$ distance computations per player construct all profiles; without reset cycles, a second algorithm processes rank changes directly. The history results extend to infinite arenas with a well-founded potential for resets and arbitrary branching.

1 Introduction

A repetition claim can change a position's value even when its vertex and remaining budget are unchanged. How much history is needed to evaluate a position and maintain its value after every legal move?

The model is motivated directly by rule-based game search. In chess, a player may claim a draw when the same position appears for the third time, including when the next announced move would create that occurrence; fivefold repetition is automatic. The same rules also use a move counter: a player may claim after 50 moves by each side without a pawn move or capture, and 75 moves are an automatic draw [8]. Pawn moves and captures reset this counter and are irreversible in the chess position graph, so a directed cycle cannot contain such a reset. Shogi has the same basic path dependence through fourfold repetition (*sennichite*) [9]. In game-tree search, transposition tables deliberately merge positions reached by different move sequences, but their repetition status can differ. Reusing a value or proof without accounting for that history is the graph-history interaction problem [10]. Our abstraction isolates this interaction while retaining a chess-like expiring counter.

Quiet moves consume reserve; resets restore a common capacity. When no reset lies on a cycle, the clock product is well-founded. A positional winning policy then works at every winning reserve and never revisits a vertex. One value query therefore needs only a fixed draw boundary in the current strongly connected component, together with any claim at the current vertex.

This boundary is not enough for online monitoring: a smaller count may become relevant after later visits. But alternation permits a vertex to occur at most once every two positions. Counts too small to reach the threshold before expiration can be discarded permanently. The remaining counts admit an encoding by bounded integer vectors of total mass at most $S = (H - 2k + 2)_+$. Equal-length live histories give the matching lower bound in Theorem 5.2.

For the clock game, both players have positional policies optimal at every entering reserve (Theorem 3.2), though not a single policy for all repetition histories. As capacity varies, a rank changes only when some vertex first becomes winning at full reserve. This bounds the number of profile changes and permits computation without enumerating capacities. Unlike the history reduction, these results allow reset cycles and positive integer costs (Theorem 7.2).

Related work. Reachability attractors and positional strategies are standard [1]. Our fixed-capacity computation uses min–max shortest-path methods [2]; Sanada et al. [3] give a coalgebraic formulation. The contribution here concerns changes across all capacities, not the distance algorithm.

Consumption games allow independently reloadable resources [4]; recharge and average-energy games allow recurring replenishment under different objectives [5]. Consumption Markov decision processes also admit polynomially represented resource-aware strategies [6]. Iterating over resets into known winning states is established in counter reachability games [7, Section 2.4]. We use it to bound and construct complete capacity profiles.

Kishimoto and Müller [10] test whether search proofs remain valid under another history. Here a static draw boundary replaces a supplied history under optional current and prospective claims. First-cycle games [11] instead terminate at the first cycle and evaluate it. Arena-independent finite-memory strategies [12] concern optimal play; our monitor must also handle moves that decline a win or claim.

The count reduction resembles frequency-counting methods [13], but is exact: it discards only information irrelevant to every remaining continuation. The randomized lower bound realizes the standard augmented-index information argument [14] through legal game continuations.

2 Arenas, clocks, and claims

An arena has $N = |V|$ nonterminal vertices and $M = |E|$ actions. Each vertex p is owned by $\mathrm{own}(p) \in \{+, -\}$ and has at least one action. An action either leads to another nonterminal or ends with outcome $+1$, 0, or -1. Player $+$ maximizes and player $-$ minimizes. Continuing actions are either resets or quiet actions. Unless stated otherwise, all actions have unit cost. The basic game has no clock and assigns outcome zero to infinite plays.

For Sections 3–6, assume that no directed cycle contains a reset. Equivalently, there is a finite-range potential Φ that is nonincreasing on continuing edges and strictly decreases on resets. This follows by topologically ordering the strongly connected components (SCCs): every reset must leave its component. Quiet edges may also leave a component.

Fix a capacity $H \geq 1$. A live clock state has reserve $r \in \{1, \dots, H\}$ and clock $c = H - r$. In the clock game Γ_H, terminal outcomes are evaluated first, a reset restores reserve H, and a quiet action to a nonterminal consumes one unit, drawing if it exhausts the reserve. A terminal victory thus succeeds on the last available action. Every continuing transition decreases $(\Phi(p), r)$ lexicographically, so backward induction defines $V_H(p, r)$.

For the history results, continuing actions also alternate owners, and a terminal victory belongs to its mover. Fix $k \geq 2$. A player may claim a draw at a vertex that has occurred at least k times, or before an action that would create a kth occurrence. Clock claims have the same current and prospective forms at H quiet moves

since the last reset. Later automatic draw thresholds may also be present. Histories may start at any vertex with clock zero and are promised live and clock-consistent, with $c < H$. A reset starts a new phase.

At clock $H - 1$, a quiet nonterminal action gives its mover a prospective clock claim and the next player a current claim. Either player can therefore prevent a loss across this boundary, so replacing the action by a draw preserves value, including when repetition claims are present. Terminal actions and resets are unchanged. The restriction on terminal victories matters: a prospective clock claim cannot improve the mover's terminal win or draw, but could improve an action that wins for the opponent. Thus Γ_H gives the exact clock boundary below expiration.

Logarithms in memory bounds are base two. Operation counts are stated separately from integer bit lengths.

3 Ranks and optimal policies

For $\sigma \in \{+, -\}$, identify its winning outcome with $+1$ or -1 and define

$$d_\sigma^H(p) = \min\{r : V_H(p, r) = \sigma\}, \qquad \min \varnothing = \infty. \tag{1}$$

A terminal victory for σ has rank zero; other terminals have rank infinity. More reserve cannot destroy a win: simulate the smaller reserve until the first reset, when the reserves coincide. Increasing the common capacity also preserves a win by simulating the smaller reserve after each reset. Neither argument assumes a positional policy.

Write $[x]_H = x$ for $x \le H$, and $[x]_H = \infty$ otherwise. For an action e at p, set

$$b_\sigma^H(e; d) = \begin{cases} 1, & e \text{ ends in a victory for } \sigma, \\ \infty, & e \text{ ends in any other outcome}, \\ 1, & e \text{ resets to } q \text{ and } d(q) < \infty, \\ \infty, & e \text{ resets to } q \text{ and } d(q) = \infty, \\ [1 + d(q)]_H, & e \text{ is quiet and leads to } q. \end{cases} \tag{2}$$

The vertex equations are

$$d(p) = \begin{cases} \min_{e \in E(p)} b_\sigma^H(e; d), & \mathrm{own}(p) = \sigma, \\ \max_{e \in E(p)} b_\sigma^H(e; d), & \mathrm{own}(p) \ne \sigma. \end{cases} \tag{3}$$

Theorem 3.1 (Ranks and winning policies)**.** *If no reset lies on a directed cycle, equations* (2)–(3) *have a unique solution in* $\{1, \dots, H, \infty\}^V$*, namely* d_σ^H*. Player* σ *wins exactly when* $r \ge d_\sigma^H(p)$*. Choosing one minimizing action at each own vertex with finite rank gives a positional policy winning from every such state, independently of the entering reserve. No vertex repeats along a play of this policy from a winning state.*

Proof. Winning reserves are upper intervals. An own choice takes their union; an opposing choice takes their intersection. Their endpoints give the equations.

For uniqueness, lift a proposed solution to $X(p, r) = [r \ge d(p)]$. The equations give the winning recurrence on the clock product. Well-founded induction on $(\Phi(p), r)$ identifies X with the true winning predicate and hence fixes its endpoints.

Every chosen own quiet action and every opposing quiet action from a finite-rank vertex satisfy $d(q) \le d(p) - 1$. Resets enter wins at full reserve. The policy preserves the winning interval and reaches a winning terminal. A repeated vertex contradicts rank descent if no reset intervenes, and potential descent otherwise. □

Theorem 3.2 (Optimal policies for every reserve)**.** *For fixed H, each player σ has a positional policy on V that is minimax optimal from every (p, r), $1 \le r \le H$, without observing r. At a σ-owned vertex, choose*

$$\pi_\sigma(p) \in \begin{cases} \arg\min_e b^H_\sigma(e; d^H_\sigma), & d^H_\sigma(p) < \infty, \\ \arg\max_e b^H_{-\sigma}(e; d^H_{-\sigma}), & d^H_\sigma(p) = \infty. \end{cases} \tag{4}$$

The two policies form a saddle point at every clock state.

Proof. A vertex cannot be winning at full reserve for both players. Any action supporting a finite σ-rank either wins immediately or reaches a vertex winning for σ at full reserve, so its requirement for $-\sigma$ is infinite. The first branch of (4) therefore also maximizes the opponent's requirement.

Let $\tau = -\sigma$. The policy preserves nonloss whenever

$$r < d^H_\tau(p). \tag{5}$$

At a σ-owned vertex the chosen τ-requirement equals the maximum $d^H_\tau(p)$; at a τ-owned vertex every requirement is at least this minimum. A quiet action expires to draw or leaves $r - 1 < d^H_\tau(q)$. A reset must reach infinite τ-rank, and a τ-winning terminal action is impossible. Thus τ cannot win.

At a σ-winning state, Theorem 3.1 gives victory. At a draw state, (5) gives nonloss. At a τ-winning state, the value is already σ's worst outcome. These cases prove optimality for both players. □

Proposition 3.3 (Computation and checking)**.** *Both rank vectors and the policies can be computed in $O((N + M)\log(N + 1))$ graph operations and $O(N + M)$ storage. A proposed rank vector can be verified in $O(N + M)$ operations on $O(\log(H + 1))$-bit integers.*

Proof. Process SCCs in reverse topological order. Reset targets and quiet exits already have known ranks. Inside a component, use min–max label setting [2]: offer an own vertex a finite outgoing requirement, and an opposing vertex its maximum once every outgoing requirement is finite. Settle the smallest offer, propagate its value plus one to quiet predecessors, and discard offers above H.

Supporting quiet successors have smaller rank. A finite unsettled vertex would therefore have a decreasing dependency chain ending at an available terminal or exit, contradicting propagation. Addressable queues process each edge a constant number of times. Choosing policies and checking (3) require one further edge scan. □

4 Evaluating a supplied history

Let a live history h end at p, and let n_h count occurrences, including the current one. Define the *repetition frontier* and the bit recording a current claim by

$$F_h = \{q : n_h(q) \ge k - 1\}, \qquad \chi_h = \mathbf{1}_{\{n_h(p) \ge k\}}. \tag{6}$$

In the static history game Γ^{F_h, χ_h}_H, redirect every future action into F_h to a draw terminal. Start at a copy p^* with p's redirected outgoing actions, no incoming actions, and an extra draw action exactly when $\chi_h = 1$. The current occurrence of p is not counted again, and F_h remains fixed.

Theorem 4.1 (History reduction)**.** *If no reset lies on a directed cycle, then under the claim conventions of Section 2,*

$$\mathrm{Val}_{\mathrm{claims}}(p, h) = \mathrm{Val}_{\Gamma^{F_h, \chi_h}_H}(p^*, H - c). \tag{7}$$

Later automatic draw thresholds do not change this equality.

Proof. A nonlosing strategy in the static game transfers to the claim game: claim prospectively when its prescribed action would enter F_h, and claim at the next vertex if the opponent enters F_h without claiming. The root claim is justified by χ_h and clock absorption by Section 2. Further repetitions only add opportunities to draw and cannot spoil nonloss.

Suppose now that the static value is a victory for σ. Any root claim belongs to σ, who can ignore it. Remove that action and identify the entry copy with p; its outgoing actions and winning status are unchanged. Theorem 3.1 gives a winning policy that never repeats a vertex or enters the frontier. Every subsequent vertex outside F_h had at most $k-2$ occurrences and gains at most one; no member of F_h, including the root, is reentered. No harmful repetition or clock claim arises, so the policy wins the claim game. When the static value is draw, transfer a nonlosing strategy for each player. □

Theorem 4.2 (Restriction to the current component)**.** *Let C be the SCC of the current vertex. Only $F_h \cap C$ affects the history correction. The clock value with a fresh history gives exact boundary data at every exit from C. Given the SCC decomposition and fresh ranks, a query takes*

$$O\big((|C| + |E_C|)\log(|C| + 1)\big)$$

graph operations, plus reading or decoding the history summary; E_C contains actions with source in C.

Proof. Every historical vertex q has a path $q \rightsquigarrow p$. If it remains reachable from p, it lies in C. After an exit, neither a historical vertex nor a vertex just visited in C can recur. A fresh winning policy cannot meet an earlier occurrence, and fresh nonloss survives added draw claims. Fresh values are therefore exact, including on quiet exits at nonzero clock. Apply Proposition 3.3 inside C with these boundary values and the redirected frontier. □

Proposition 4.3 (Frontier size)**.** *For a clock-consistent entering history,*

$$|F_h \cap C| \le \left\lfloor \frac{\lceil (c+1)/2 \rceil}{k-1} \right\rfloor + \left\lfloor \frac{\lfloor (c+1)/2 \rfloor}{k-1} \right\rfloor. \tag{8}$$

For every $k \ge 2$ and $c \ge 0$, a finite alternating arena without resets has a live history with $H > c$, $\chi_h = 0$, and equality.

Proof. No vertex preceding the last reset can recur, since the potential strictly decreased. The current phase has $c+1$ occurrences, split between owner classes of sizes $\lceil (c+1)/2 \rceil$ and $\lfloor (c+1)/2 \rfloor$. Each frontier vertex uses at least $k-1$ occurrences in its class.

For equality, take a complete bipartite arena. In each parity subsequence, assign blocks of $k-1$ occurrences to distinct vertices and any remainder to one more vertex. Interleaving gives a legal history in one SCC, with all counts at most $k-1$ and equality in (8). Add an unused vertex of the other owner for a one-vertex history. □

Proposition 4.4 (Failure with reset cycles)**.** *With a reset on a cycle, histories with the same root, clock, empty frontier, and no current claim can have different values, already for $k = 3$.*

Proof. Take $H = 4$, a $+$-winning terminal T, and the edges

$$s^- \to q^+ \to p^-, \qquad t^- \to u^+ \to p^-, \qquad p^- \to q^+,$$
$$q^+ \xrightarrow{\text{reset}} v^- \to q^+, \qquad q^+ \to a^- \to b^+ \to T.$$

The histories (s, q, p) and (t, u, p) have clock two, empty frontier, and no current claim. The forced move to q leaves reserve one, so the three-action win requires a reset. In the first history, returning from v would visit q a third time, allowing $-$ to claim. In the second, this is only the second visit, leaving reserve three and a win. Quiet looping or taking the winning path too early cannot improve the first value beyond draw. □

Remark 4.5 (A claim at the current vertex)**.** For $k \geq 3$, χ_h can matter even when root, clock, and frontier agree. Let r^- choose among $x_1^+, \ldots, x_{k-1}^+$, each of which can return to r or win immediately, and add $u^- \to x_1$. Compare

$$(r, x_1, r, x_2, r, \ldots, x_{k-1}, r), \quad (u, x_1, r, x_2, r, \ldots, x_{k-1}, r).$$

Both frontiers are $\{r\}$. With at least two reserve units, the first history draws by a current claim; the second loses for $-$. For $k = 2$, a current claim is redundant: an earlier departure from the repeated root gives an action to an already visited vertex and hence a prospective claim.

5 Memory bounds

Definition 5.1. An exact value monitor is a deterministic machine updated by each legal action. It must report the claim-game value on every promised live history with $c < H$. The arena, H, and k are fixed read-only inputs. The current vertex and clock are separately available, as are the old state and action during an update. Memory counts persistent history bits; temporary workspace cannot retain the past, and the input stream cannot be reread. The monitor need not detect automatic termination or process a terminated history.

Let $\mathcal{M}(N, H, k)$ be the maximum, over N-vertex alternating arenas with no reset on a cycle, of the minimum persistent history memory required by such a monitor.

The monitor must handle every legal continuation, including moves that decline an immediate win or an optional claim. Its memory is not the memory of an optimal strategy.

For $n, K, L \in \mathbb{N}$, let

$$\mathcal{U}(n, K, L) = \left| \left\{ z \in \{0, \ldots, K\}^n : \sum_i z_i \leq L \right\} \right|, \tag{9}$$

so $\mathcal{U}(n, K, 0) = 1$.

Theorem 5.2 (Deterministic memory)**.** *Put $S = (H - 2k + 2)_+$. For $N \geq 8$, with absolute constants in the asymptotic bounds:*

(i) If $H \leq 2k - 2$, then $\mathcal{M}(N, H, k) = 0$.

(ii) If $k = 2$ and $H \in \{3, 4\}$, then $\mathcal{M}(N, H, k) = 1$.

(iii) In every other case with $H \geq 2k - 1$,

$$\mathcal{M}(N, H, k) = \Theta\big(\log \mathcal{U}(N, k - 1, S)\big) \tag{10}$$

$$= \Theta\left(\min \left\{ N \log k,\ \log \binom{N + S}{S} \right\} \right). \tag{11}$$

For every N and $H \geq 2k - 1$, an upper bound is

$$\lceil \log \mathcal{U}(N, k - 1, S) \rceil + 1. \tag{12}$$

Here N may be replaced by the largest SCC size. The lower bounds use arenas without resets and place all nontrivial history information in one SCC.

We first encode the counts that can still affect value, then construct distinguishable histories. The cases $k = 2$, $H \in \{3, 4\}$ require a separate argument: (11) would incorrectly give logarithmic memory there.

5.1 Discarding irrelevant counts

For owner $i \in \{+, -\}$, let $A_i(c)$ count its positions among the $c + 1$ occurrences in the current phase. Let Q_i count its positions in the entire live phase $0, \dots, H - 1$. The two Q_i are $\lceil H/2 \rceil$ and $\lfloor H/2 \rfloor$; their assignment follows from the current owner and the parity of c, without extra memory.

Set $B_i = Q_i - k$. If $B_i < 0$, this class cannot reach a kth live occurrence. Otherwise define

$$\eta_i(c) = \max\{1, A_i(c) - B_i\}. \tag{13}$$

Discard counts from earlier phases and put $\widehat{n}(q) = \min\{n_h(q), k - 1\}$. Retain a positive count exactly when

$$B_{\mathrm{own}(q)} \geq 0, \qquad \widehat{n}(q) \geq \eta_{\mathrm{own}(q)}(c). \tag{14}$$

Store χ_h separately.

Lemma 5.3 (Updating the retained counts)**.** *The retained counts and χ_h can be updated exactly after each legal action. An omitted positive count never becomes retained later in the same phase. The retained frontier and χ_h give the same value as the full frontier in Theorem 4.1.*

Proof. When $B_i \geq 0$ and $\eta_i = 1$, an omitted count is zero. Once $\eta_i > 1$, each visit to class i increases both A_i and η_i by one. The count at the visited vertex increases by at most one, so an omitted positive count cannot catch the threshold. Other visits cannot close the gap. At a reset, discard all earlier counts, since no earlier vertex can recur.

Before a move into q below expiration, its class satisfies $A_i \leq Q_i - 1$. If $B_i \geq 0$, this gives $\eta_i \leq k - 1$, so any old count of $k - 1$ is retained. If $B_i < 0$, a kth occurrence is impossible. Thus the new claim bit is exactly the predicate that the old capped count at q equals $k - 1$. Increment retained counts, add an unseen vertex when its threshold is one, and apply the new thresholds. No omitted positive count must be recovered.

A frontier member can be erased only when $\eta_i > k - 1$. This forces $A_i = Q_i$, leaving no further live occurrence in its class. It cannot be reentered before expiration or after a reset. Other erased counts were below $k - 1$ and hence outside the frontier. The remaining frontier therefore gives the same future draw boundary, while χ_h handles a claim at the current vertex. □

Lemma 5.4 (Bound on the stored counts)**.** *Define*

$$z(q) = \begin{cases} \widehat{n}(q) - \eta_i(c) + 1, & q \text{ is retained and } \mathrm{own}(q) = i, \\ 0, & q \text{ is not retained.} \end{cases} \tag{15}$$

Then $0 \leq z(q) \leq k - 1$ and

$$\sum_q z(q) \leq (B_+ + 1)_+ + (B_- + 1)_+ = S. \tag{16}$$

Proof. A class with $B_i < 0$ contributes zero. If $\eta_i = 1$, its capped counts sum to at most $A_i \leq B_i + 1$. If $\eta_i > 1$ and $s \geq 1$ vertices are retained, then

$$\sum_{\mathrm{own}(q)=i} z(q) \leq A_i - s(\eta_i - 1) = B_i + 1 - (s - 1)(\eta_i - 1) \leq B_i + 1.$$

An empty class contributes zero. Sum over the classes and substitute their parity counts Q_i to obtain (16). □

An enumerative code stores z in $\lceil \log \mathcal{U}(N, k-1, S) \rceil$ bits, plus one bit for χ_h. The current vertex and clock determine the thresholds, so positive coordinates recover the retained counts. Decoding and updating use temporary workspace; no update-time bound is asserted for this encoding.

If $H \le 2k-2$, neither class has k live occurrence slots. No repetition claim can affect play before expiration, and no earlier phase can contribute. Fresh values are exact, proving part (i). For the SCC bound, discard counts when leaving their component and encode coordinates relative to the current SCC. The clock still determines A_i, so this needs no further counter.

5.2 The lower bound for $S \ge 3$

Lemma 5.5 (Distinguishable histories)**.** *For $m \ge 1$, $k \ge 2$, and $1 \le L \le m(k-1)$, an alternating arena without resets has $6m+2$ nonterminals and, for every*

$$H \ge 2L + 2k - 1, \tag{17}$$

$\mathcal{U}(m, k-1, L)$ equal-length live histories at the same root and clock, pairwise distinguishable by common legal continuations. Only player + has nontrivial choices, and the arena is one SCC.

Proof. Use $2m$ vertices owned by +: data vertices $x_1, \dots, x_m$ and m padding vertices. Each has a private connector owned by − whose only action enters that vertex. Every data or padding vertex can enter every connector, move to a common vertex d^- with forced successor r^+, or win immediately at a terminal T.

For each x_j, add vertices s_j^-, p_j^-, each with only successor x_j, and actions from both r and x_j to each. The root r also wins immediately at T. This gives $2m$ data and padding vertices, $2m$ connectors, $2m$ added vertices, and d, r: $6m+2$ in total. The root reaches every vertex through the data vertices and connectors, and every vertex returns through d, r, so the arena is one SCC.

For each $n \in \{0, \dots, k-1\}^m$ with $\sum n_j \le L$, assign the padding vertices counts at most $k-1$ totaling $L - \sum n_j$. Fix such an assignment, which exists because $L \le m(k-1)$. Order all L occurrences canonically. Start at the first vertex, use the destination's connector before every subsequent occurrence, and end with d, r. This history has $2L$ edges. No data or padding vertex occurs more than $k-1$ times, no connector occurs more often than its target, and d, r occur once. All prefixes are live without a current claim.

For distinct vectors with $n_j < n'_j$, set $t = k - 1 - n'_j \in \{0, \dots, k-2\}$ and append

$$r(s_j x_j)^t p_j, \tag{18}$$

without recounting the current r. The continuation has $2t+1$ edges. At its end, x_j has count $k-1$ in the higher-count history and at most $k-2$ in the other; p_j is fresh, and s_j has occurred at most $k-2$ times. Bound (17) leaves reserve at least two.

At p_j, player − can claim before entering x_j exactly in the higher-count history, which draws because − has no terminal victory. In the other history the forced move reaches x_j, where + wins immediately. Every continuation prefix has counts below k, so later automatic thresholds do not affect the distinction.

Equal monitor states would remain equal under this common continuation, at the same vertex and clock, and give the same output. The histories therefore require at least $\mathcal{U}(m, k-1, L)$ persistent states. □

For $S \ge 3$, choose

$$m = \left\lfloor \frac{N-2}{6} \right\rfloor, \qquad L = \min\left\{ \left\lfloor \frac{S-1}{2} \right\rfloor,\ m(k-1) \right\}. \tag{19}$$

For $N \ge 8$, $m \ge 1$ and the construction fits in N vertices; add unused vertices as needed. Since $H = 2k-2+S$, (17) holds.

Lemma 5.6 (Comparison of parameters)**.** *With* (19)*,* $N \geq 8$*, and* $S \geq 3$*,*

$$\log \mathcal{U}(N, k-1, S) \leq 52 \log \mathcal{U}(m, k-1, L). \tag{20}$$

Proof. Put $L_0 = \lfloor (S-1)/2 \rfloor$. Since $N \leq 6m + 7 \leq 13m$, partition the coordinates into at most thirteen blocks of size at most m. Also $S \leq 4L_0$. Greedily split each block into at most four nonnegative integer vectors of mass at most L_0, without increasing any coordinate. Pad each to m coordinates. Summing recovers the block, giving an injection into 52 vectors counted by $\mathcal{U}(m, k-1, L_0)$. Replacing L_0 by $\min\{L_0, m(k-1)\} = L$ does not change the count. Take logarithms. □

5.3 The capacities $2k-1$ and $2k$

Lemma 5.7. *For* $k \geq 3$ *and* $H \in \{2k-1, 2k\}$*, an* $(m+1)$*-vertex arena has* m *pairwise distinguishable equal-length histories with empty initial frontier and no current claim.*

Proof. Take $x_1^+, \ldots, x_m^+$ and p^-. Each x_j can enter p or win immediately; p can enter every x_j. The prefixes (x_i, p) have clock one and empty frontier.

Append $(x_j, p)^{k-2}$ to distinguish prefix j from any prefix $i \neq j$. At the final p, its count is $k-1$ and the clock is $2k-3$, leaving reserve at least two. Under prefix j, x_j has count $k-1$, allowing $-$ a prospective draw. Otherwise x_j has count $k-2$ and the original vertex has count one, so no $+$-owned vertex reaches $k-1$. Every move from p then lets $+$ win immediately. □

Taking $m = N-1$ gives $\log(N-1)$ bits. Since $\log \mathcal{U}(N, k-1, S) = \Theta(\log N)$ for $S = 1, 2$, this proves the lower bound for $k \geq 3$.

Lemma 5.8 (The case $k = 2$)**.** *For* $k = 2$ *and* $H \in \{3, 4\}$*, one persistent bit suffices on every arena, and some three-vertex arena requires it.*

Proof. A current claim is redundant by Remark 4.5. At reserve one, every quiet nonterminal action already draws, and a reset cannot revisit earlier vertices, so history has no effect. At clock zero the history is fresh. At clock one only the previous and current vertices have occurred; both are available during the last update, so the value is computable without earlier persistent information.

The remaining case is $H = 4$, $c = 2$, $r = 2$. Before expiration, a quiet move can revisit only the immediately previous vertex. An older vertex of the current owner's class requires two moves, reaching the clock boundary; resets cannot revisit it. The last action therefore determines the relevant frontier. A repeated current vertex adds no effect: the earlier departure giving its prospective claim already leads to that previous vertex.

Adding draw claims can preserve a fresh value or change it to zero, but cannot reverse its sign. Store one bit indicating whether it changed. The current vertex and clock determine the fresh value. The preceding cases update this bit without retaining the previous vertex.

For necessity, take x^+, y^+, p^- and actions $x \to p$, $x \to T$, $y \to p$, $p \to x$, where T wins for $+$. Histories (x, p) and (y, p) have the same root and clock. The first draws by a prospective claim; in the second, $p \to x \to T$ wins before expiration. Two persistent states are necessary. □

5.4 Counting the possible encodings

Lemma 5.9. *For* $N, K, S \geq 1$*,*

$$\log \mathcal{U}(N, K, S) = \Theta\left(\min\left\{N \log(K+1),\ \log \binom{N+S}{S}\right\}\right) \tag{21}$$

with absolute constants.

Proof. The upper bounds count all $(K+1)^N$ bounded vectors or all nonnegative integer vectors of mass at most S. For $N = 1$, $\mathcal{U}(1, K, S) = 1 + \min(K, S)$ gives the result directly; assume $N \geq 2$.

If $S \leq N/2$, binary vectors with exactly S ones give a logarithm of order $S \log(1 + N/S)$, matching the unbounded count. If $N/2 < S \leq N$, binary vectors with at most $\lfloor N/2 \rfloor$ ones give order N, and the upper bound is $O(N)$.

For $S \geq N$, the cube with coordinates in

$$\{0, \ldots, \min(K, \lfloor S/N \rfloor)\}$$

has mass at most S and logarithmic size of order $N \log(1 + \min\{K, S/N\})$. The inequality $\binom{a}{b} \leq (ea/b)^b$, with $b = N$, bounds the smaller upper bound by a constant multiple of this expression. □

Lemmas 5.3–5.9 prove Theorem 5.2. Away from its two exceptions,

$$\mathcal{M}(N, H, k) = \begin{cases} \Theta\big(S \log(1 + N/S)\big), & 1 \leq S \leq N, \\ \Theta\big(N \log(1 + \min\{k-1, S/N\})\big), & S \geq N. \end{cases}$$

For $S \geq N(k-1)$, the mass restriction disappears, giving $\Theta(N \log k)$. Storing all capped counts uses $N \lceil \log k \rceil + 1$ bits and avoids enumerative coding.

Even empty frontiers can conceal different future values. For $k \geq 3$, restrict data and padding counts in Lemma 5.5 to $\{0, \ldots, k-2\}$ and take $L = m(k-2)$. This gives $(k-1)^m$ equal-length histories with empty frontier and $\chi_h = 0$, distinguished by the same continuations. Every other vertex in a writing history occurs once, below $k-1$.

6 Randomized memory in the dense regime

A randomized monitor has error at most δ if it reports the wrong value with probability at most δ on each fixed live history. This is a per-query guarantee, not simultaneous correctness on all histories. We allow public random coins. The following games have optional claims but no automatic repetition threshold, so continuations may decline available claims.

Theorem 6.1 (Randomized lower bound)**.** *Fix $0 \leq \delta < 1/2$. On the $6m + 2$ vertex arena of Lemma 5.5, with*

$$H \geq 2m(k-1) + 2k - 1,$$

every randomized value monitor with error at most δ needs at least

$$\big(1 - \mathsf{h}_2(\delta)\big) m \lfloor \log k \rfloor \tag{22}$$

persistent bits, where h_2 is binary entropy. For $k \geq 3$, histories with identical empty initial frontier give the analogous bound with $\lfloor \log(k-1) \rfloor$ and capacity $H \geq 2m(k-2) + 2k - 1$.

Proof. Use all digits $n_j \in \{0, \ldots, k-1\}$ and pad to total $L = m(k-1)$, giving equal-length writing histories. To test $n_j \geq a$, for $a \in \{1, \ldots, k-1\}$ append

$$r(s_j x_j)^{k-1-a} p_j.$$

At p_j the value is draw exactly when $n_j \geq a$, and a +-victory otherwise; reserve is at least two. A high digit may cause a kth occurrence along the continuation, but declining the claim is legal. Player − has no terminal victory and can claim at the final test precisely when the inequality holds.

Put $b_0 = \lfloor \log k \rfloor$ and restrict each digit to $\{0, \ldots, 2^{b_0} - 1\}$. Let X be the resulting mb_0 independent uniform bits, ordered by digit, most significant bit first. Given a digit's preceding bits, the threshold with that prefix followed by one and then zeros tests its next bit. Thus the stored state Z, public randomness R, and preceding bits decode each next bit with error at most δ. This is the augmented-index argument [14].

For each bit, the decoder's error determines the true bit, giving

$$H(X_\ell \mid Z, R, X_{<\ell}) \leq \mathsf{h}_2(\delta).$$

The entropy chain rule yields

$$H(X \mid Z, R) \leq mb_0 \mathsf{h}_2(\delta), \qquad I(X; Z \mid R) \geq mb_0(1 - \mathsf{h}_2(\delta)).$$

Since R is independent of X and b stored bits permit at most 2^b states, $I(X; Z \mid R) \leq H(Z \mid R) \leq b$, proving (22).

For empty initial frontiers, restrict data and padding counts to $0, \ldots, k-2$ and take $L = m(k-2)$. All writing counts are below $k - 1$. Encode $\lfloor \log(k-1) \rfloor$ bits per digit and apply the same tests. □

This matches the deterministic bound in the dense regime, but does not establish a randomized counterpart of the full curve in Theorem 5.2.

7 Capacity profiles

Here we consider clock games without repetition claims and allow arbitrary reset cycles, without requiring alternation or assigning terminal victories to the mover. Actions have positive integer costs $w(e)$, with $W = \max_e w(e)$. If $r < w(e)$, the action draws. Otherwise a terminal action succeeds and a reset restores H. A quiet nonterminal action draws at $r = w(e)$ and leaves $r - w(e)$ when $r > w(e)$. Infinite plays draw. Unit costs recover the earlier clock game.

Budget monotonicity still holds. For player σ, let d_σ^H be its least winning reserve and let

$$W_H^\sigma = \{p : d_\sigma^H(p) < \infty\}$$

be the vertices winning at full reserve. We count maximal intervals on which a profile $H \mapsto d_\sigma^H(p)$ is constant and finite, excluding any initial infinite segment.

7.1 Winning sets and breakpoints

For $X \subseteq V$, replace each reset to q by a terminal victory for σ if $q \in X$, and a draw otherwise, keeping its cost. Other actions are unchanged. Let $\ell_X(p)$ be the least reserve that forces success in this game. This is a positive-cost min–max distance, with infinite plays unsuccessful. A finite distance is at most NW: a winning shortest-path policy strictly decreases distance and uses at most N actions. Proposition 3.3's label-setting procedure computes these distances with costs in place of unit increments.

Lemma 7.1 (The winning set as a least fixed point)**.** *For each H and σ, W_H^σ is the least fixed point of*

$$T_H(X) = \{p : \ell_X(p) \leq H\}. \tag{23}$$

Moreover,

$$d_\sigma^H(p) = \begin{cases} \ell_{W_H^\sigma}(p), & \ell_{W_H^\sigma}(p) \le H, \\ \infty, & \ell_{W_H^\sigma}(p) > H. \end{cases} \tag{24}$$

Proof. Iterate the monotone map T_H from $X_0 = \varnothing$, with $X_{j+1} = T_H(X_j)$. A vertex in X_1 wins without resetting; one in X_{j+1} can first reset to a vertex already winning at full reserve in X_j. Composing these strategies shows that every iterand consists of genuine wins.

Conversely, a winning attractor strategy in the finite clock product reaches a terminal within a bounded number of transitions, hence with a bounded number of resets. Induction on that bound puts its starting vertex at full reserve in some X_j. The increasing sequence stabilizes after at most N strict increases, and its union is W_H^σ, the least fixed point.

A win from reserve $r \le H$ must first reach a winning terminal or reset to W_H^σ. Conversely, a strategy achieving either within reserve r composes with a winning strategy after the reset. Minimizing r gives (24). □

Theorem 7.2 (Number of profile changes)**.** *On every finite positive-cost arena, including those with reset cycles:*

(i) every finite rank is at most NW, and every profile stabilizes for $H \ge NW$;

(ii) every profile breakpoint is a capacity at which some vertex first becomes winning at full reserve for one player;

(iii) across both players, there are at most N distinct breakpoint capacities and at most $N(N+1)/2$ finite constant runs.

The quadratic order is attained by unit-cost acyclic alternating arenas with only one choosing player and at most one reset per play.

Proof. The distance bound gives finite ranks at most NW. In the basic game, a positional attractor strategy reaches a winning terminal in at most N actions, so capacity NW admits it from full reserve. A clock win is also a basic win, since removing expiration cannot hurt a strategy forcing its terminal. Both winning sets therefore stabilize by NW, and (24) fixes the ranks thereafter.

While W_H^σ is unchanged, $\ell_{W_H^\sigma}$ is fixed and every already finite rank is constant. Any new finite rank itself enlarges the winning set. Each breakpoint therefore coincides with some vertex first becoming winning. No vertex can become winning for both players, even at different capacities, because winning status is monotone. There are at most N such capacities.

Order them as $t \le N$ events, and let a_j count vertices that have become winning by event j. At most a_j finite runs begin there. At least $t - j$ distinct vertices become winning later, so $a_j \le N - (t - j)$. Thus

$$\sum_{j=1}^{t} a_j \le tN - \frac{t(t-1)}{2} \le \frac{N(N+1)}{2}.$$

Proposition 7.5 gives the quadratic lower bound; sharpness of the triangular constant is not claimed. □

Theorem 7.3 (Constructing all profiles)**.** *All profiles on an arena with arbitrary resets and positive costs can be computed with at most $N + 1$ distance computations per player, using*

$$O\big(N(N+M)\log(N+1)\big)$$

arithmetic operations and $O(N + M + K)$ storage, where K is the number of output runs. Finite integers have $O(\log(NW + 1))$ bits. The algorithm does not enumerate capacities through NW.

Algorithm 1 Capacity profiles for a fixed player

$X \leftarrow \varnothing$; compute ℓ_X; initialize empty run lists
while $\min_{p \notin X} \ell_X(p) < \infty$ **do**
  $h \leftarrow \min_{p \notin X} \ell_X(p)$
  repeat
    $A \leftarrow \{p \notin X : \ell_X(p) \le h\}$
    if $A \neq \varnothing$ **then**
      $X \leftarrow X \cup A$; recompute ℓ_X
    end if
  until $A = \varnothing$
  for $p \in X$ whose recorded rank differs from $\ell_X(p)$ **do**
    append $(h, \ell_X(p))$ to the run list at p
  end for
end while

Proof. Starting from $X = \varnothing$, Algorithm 1 computes ℓ_X. The next possible breakpoint is

$$h = \min_{p \notin X} \ell_X(p).$$

At this capacity it repeatedly adds vertices with distance at most h and recomputes, recording ranks only after the set stops growing.

By Lemma 7.1, the resulting set is W_h^σ. Before the next proposed h, no outside vertex enters $T_H(X)$, so X remains a fixed point. Monotonicity from the previous least fixed point makes it the least one. No breakpoint is skipped, and recording only after closure avoids intermediate values that are not profile values.

Each recomputation after initialization adds at least one vertex, giving at most $N + 1$ distance computations in total. Each costs $O((N + M) \log(N + 1))$ operations. Recording at most N events adds $O(N^2)$ work, within the bound. The graph and working distances use linear storage, plus the output. □

Leastness is essential. On a cycle containing only resets and no winning terminal, finite local ranks can satisfy the reset equations even though every play draws. Iteration from $\varnothing$ never declares such a cycle winning. Reset cycles can also prevent a positional policy from winning at every reserve. With $H = 4$ and

$$q^+ \xrightarrow{\text{reset}} v^- \to q^+, \qquad q^+ \to a^- \to b^+ \to T,$$

player + wins from $(q, 1)$ by resetting once, then taking the three-action path. Always resetting loops; always taking the path expires from reserve one. Thus the profile theorems apply more broadly than Theorem 3.2.

7.2 Processing rank changes directly

When no reset lies on a cycle, the weighted rank and policy theorems follow by replacing unit requirements with $w(e)$ and quiet requirements with $[w(e) + d(q)]_H$. Quiet ranks strictly decrease, and the clock product remains well-founded.

For player σ, let $K_\sigma(q)$ count the finite runs at q, and put

$$P_\sigma = \sum_{q \in V} \deg_{\mathrm{q}}^-(q)\, K_\sigma(q), \tag{25}$$

where $\deg_{\mathrm{q}}^-$ counts incoming quiet actions.

Theorem 7.4. *On a positive-cost arena with no reset on a cycle, all profiles for σ can be constructed in*

$$O\big((N + M + P_\sigma)\log(N + M + 1)\big) \tag{26}$$

arithmetic operations and $O(N + M + K_\sigma)$ storage, where $K_\sigma = \sum_q K_\sigma(q)$. Each incoming quiet edge is processed once per successor run; each incoming reset edge is activated at most once.

Proof. Number SCCs in reverse topological order, so their index κ is nonincreasing along edges and strictly decreases on resets. Initially every published rank is infinity. Store each edge's untruncated requirement: $w(e)$ for a winning terminal; infinity for another terminal; $w(e) + d(q)$ for a quiet edge; and $w(e)$ for a reset once its target has finite published rank, infinity beforehand. Treat values above NW as infinity.

At each vertex take the minimum requirement for an own choice and the maximum otherwise. Whenever its finite aggregate b improves its published rank, place or replace the event

$$\big(\max\{h_{\text{current}}, b\},\ \kappa(p),\ b,\ p\big) \tag{27}$$

in an addressable priority queue. Process events in lexicographic order. Publish the proposed rank and its capacity, update incoming quiet requirements, and update incoming reset requirements only when the rank first becomes finite.

Each published rank certifies a win at that capacity: supporting quiet successors have smaller published ranks, and reset targets lie in lower SCCs already justified at no greater capacity. Later decreases preserve the certificate. Published ranks are therefore never below the true ranks.

Fix a capacity h and order the true finite improvements not yet published by $(\kappa(p), d^h_\sigma(p))$. A minimal missing improvement has its required dependencies in lower components and supporting internal quiet dependencies already correct, since internal quiet successors have smaller true ranks. One supporting successor suffices at a minimizing vertex; every successor supports the rank at a maximizing vertex. Its aggregate is therefore the true rank, queued at h. Any proposed rank that is too large would be preceded by a missing improvement in a lower component or of smaller rank in the same component, contradicting queue order. Hence each publication is final for that capacity, and none of its improvements remain missing after the events are exhausted. Exactly the maximal finite runs are output.

Initialization scans the edges. Quiet requirements change once per successor run, giving P_σ updates. Reset requirements change only when their targets first become winning, giving at most M updates. Use addressable heaps for outgoing extrema and one pending event per vertex. Each operation costs $O(\log(N + M + 1))$, and each publication is charged to initialization or a requirement change. This gives the time and storage bounds. □

The bound charges each quiet edge separately for changes at its target; it is not an $O(M + K_\sigma)$ bound. Capacities with no rank changes require no work.

7.3 A quadratic example

Proposition 7.5. *There are unit-cost acyclic alternating arenas with $N = 4m - 1$ nonterminals and $\Theta(N^2)$ finite profile runs. Only player + makes choices, and every play uses at most one reset.*

Proof. Take the forced chain $z_{2m} \to \cdots \to z_1 \to T$, where T wins for + and odd- and even-indexed z_j belong to + and −, respectively. Add $x^+_0, \ldots, x^+_{m-1}$ with resets

$$x_i \xrightarrow{\text{reset}} z_{2(m-i)}.$$

For $i < m - 1$, add the quiet alternative $x_i \to y^-_i \to x_{i+1}$. There are $2m + m + (m - 1) = 4m - 1$ vertices.

Put $u = m - i$ and $a = \lfloor H/2 \rfloor$. The earliest usable reset is $\max(0, u - a)$ upper stages away, so

$$d_+^H(x_i) = 1 + 2\max(0, u - a) \tag{28}$$

when $a \geq 1$ and the right-hand side is at most H; otherwise it is infinity. The finite ranks are the distinct odd values obtained for

$$\lceil u/2 \rceil \leq a \leq u,$$

with truncation deciding whether the first feasible capacity is odd or even. There are $\lfloor u/2 \rfloor + 1$ such runs. Summing over $u = 1, \ldots, m$ gives $\Theta(m^2) = \Theta(N^2)$. Every reset enters the forced chain, which has no further reset. □

8 Infinite arenas

For an infinite arena, absence of reset cycles is not enough. Instead, assume a map $\Phi : V \to \mathcal{W}$ into a well-founded partially ordered set, with $\Phi(q) \leq \Phi(p)$ on continuing edges and strict inequality on resets. The arena may be any set-sized graph.

Theorem 8.1. *Under this potential assumption, for finite H and arbitrary branching, the unit-cost clock values and unique ranks exist. Theorems 3.1 and 3.2 hold, as do history reduction, component locality, and* (8) *for every finite live entering history. No terminating global labeling algorithm is asserted.*

Proof. The order $(\Phi(p), r)$ is well-founded: an infinite descent would either decrease Φ strictly infinitely often or eventually decrease only the positive integer reserve. Well-founded recursion therefore defines the minimax values.

Successor values lie in $\{-1, 0, 1\}$ and requirements in $\{1, \ldots, H, \infty\}$, so nonempty extrema are attained even with infinite branching. Set-theoretic choice selects policy actions. Budget simulation, uniqueness by the lifted predicate, and the nonloss invariant (5) apply unchanged. Winning plays decrease rank on quiet edges and potential on resets, reaching a terminal without repeating a vertex.

History reduction uses these winning policies and nonloss transfer. Locality uses two finite paths witnessing membership in the current SCC, not a finite component decomposition. The frontier bound counts the finite entering phase. None requires finite branching. □

Proposition 8.2 (Computability at a fixed root)**.** *Suppose the arena is finitely branching, each vertex has a computable complete finite successor list, and the potential assumption is promised. A clock value at a specified root and finite reserve is computable by finite unfolding. Every basic-game win is a win from full reserve at some finite capacity, and each root's capacity profile eventually stabilizes. There need not be a common stabilization capacity for the arena.*

Proof. The rooted clock-game tree is finitely branching with no infinite branch, hence finite: otherwise repeatedly choosing a child with infinitely many descendants would give an infinite branch. Complete successor generation makes recursive evaluation terminate, though the potential promise supplies no computable complexity bound by itself.

The same argument bounds a basic winning strategy's play lengths by some finite D. Capacity D admits it from full reserve. Once finite, the rank is a nonincreasing positive integer and changes only finitely often. A disjoint union of increasingly long forced paths rules out a common stabilization bound. □

An infinite forced chain of reset edges is acyclic but has no such potential. Its only play draws, yet constant ranks one and infinity both satisfy the local reset equations. Infinite branching causes a different failure: let a

−-owned root choose among quiet forced paths of unbounded finite length, each ending in a +-victory. The constant potential is valid and every basic play wins for +, but for each finite capacity a sufficiently long path draws. The value theorem still applies; the finite-capacity saturation conclusion does not.

9 Verification

The Python artifact uses the standard library and random seed `20260906`. It compares explicit clock-product minimax, reachability attractors allowing reset cycles, scalar ranks, occurrence-count evaluation, and both profile algorithms. The monitor is checked against full counts, not a copy of its discard rule.

Check	Comparisons or instances
Unit-cost random arenas	300
Unit-cost rank/profile entries	33,296
Unit-cost policy checks	180,272
History value contexts / encoded payloads	10,904 / 326
Exhaustive prefixes, including declined claims	81,780
Encoded histories / separating pairs	337 / 2,503
Occurrence-count evaluations of separating continuations	5,006
Exhaustive three-vertex arenas for $k = 2$, $H = 3, 4$	735
Corresponding value contexts	11,722
Queries at capacities $2k - 1$ and $2k$	1,260
Dense threshold tests / tests with empty frontiers	380 / 380
Tests of current claims	14
Weighted arenas without reset cycles	180
Weighted rank/profile entries	72,056
Weighted policy checks	1,680,114
Weighted alternating arenas with arbitrary resets / entries	240 / 77,420
Additional arenas with unrestricted ownership / entries	120 / 39,514
Arenas compared by both profile algorithms	120
Quadratic examples / profile entries	18 / 33,060

Table 1: Finite checks from the accompanying artifact. All comparisons passed. Rows overlap and are not independent statistical trials.

The unrestricted suite allows self-loops, consecutive vertices of one owner, and terminal victories for the opponent (Section 7). Weighted tests enumerate capacities through $NW + 2$ and check the profile bounds. Encoding tests check every pair's separating continuation for small bounded vectors. For $k = 2$, the suite exhausts all nonempty action subsets on two +-vertices and one −-vertex, with the permitted quiet and terminal actions. These implementation checks are neither formal proofs nor evidence for infinite-arena claims.

Without reset cycles, rank vectors and selected actions give certificates checkable by a linear edge scan. For a fixed policy, the quiet inequalities are ordinary difference constraints. The artifact includes the manuscript source, executable checks, machine-readable results, and the earlier source audit.

References

[1] T. Brihaye, A. Goeminne, J. C. A. Main, and M. Randour. Reachability games and friends: A journey through the lens of memory and complexity. In *FSTTCS 2023*, LIPIcs 284, 1:1–1:26, 2023. 10.4230/LIPIcs.FSTTCS.2023.1.

[2] M. Bardi and J. P. Maldonado López. A Dijkstra-type algorithm for dynamic games. *Dynamic Games and Applications*, 6(3):263–276, 2016. 10.1007/s13235-015-0156-0.

[3] T. Sanada, Y. Montacute, K. Phalakarn, and I. Hasuo. A coalgebraic Dijkstra algorithm. In *CONCUR 2026*, LIPIcs 391, 5:1–5:16, 2026. 10.4230/LIPIcs.CONCUR.2026.5.

[4] T. Brázdil, K. Chatterjee, A. Kučera, and P. Novotný. Efficient controller synthesis for consumption games with multiple resource types. In *CAV 2012*, LNCS 7358, pages 23–38, 2012. 10.1007/978-3-642-31424-7_8.

[5] K. G. Larsen, S. Laursen, and M. Zimmermann. Limit your consumption! Finding bounds in average-energy games. In *QAPL 2016*, EPTCS 227, pages 1–14, 2016. 10.4204/EPTCS.227.1.

[6] F. Blahoudek, T. Brázdil, P. Novotný, M. Ornik, P. Thangeda, and U. Topcu. Qualitative controller synthesis for consumption Markov decision processes. In *CAV 2020*, LNCS 12225, pages 421–447, 2020. 10.1007/978-3-030-53291-8_22.

[7] J. Reichert. *Reachability Games with Counters: Decidability and Algorithms*. PhD thesis, École Normale Supérieure de Cachan, 2015. https://lsv.ens-paris-saclay.fr/Publis/PAPERS/PDF/reichert-phd15.pdf.

[8] FIDE. *FIDE Laws of Chess Taking Effect from 1 January 2023*. Articles 9.2, 9.3, and 9.6. https://handbook.fide.com/chapter/E012023.

[9] Japan Shogi Association. *Customs of Shogi*. Advanced Rules: Draws. https://www.shogi.or.jp/event/english-pamphlet.pdf.

[10] A. Kishimoto and M. Müller. A general solution to the graph history interaction problem. In *AAAI 2004*, pages 644–649, 2004. https://cdn.aaai.org/AAAI/2004/AAAI04-102.pdf.

[11] B. Aminof and S. Rubin. First-cycle games. *Information and Computation*, 254:195–216, 2017. 10.1016/j.ic.2016.10.008.

[12] P. Bouyer, S. Le Roux, Y. Oualhadj, M. Randour, and P. Vandenhove. Games where you can play optimally with arena-independent finite memory. In *CONCUR 2020*, LIPIcs 171, 24:1–24:22, 2020. 10.4230/LIPIcs.CONCUR.2020.24.

[13] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, 1982. 10.1016/0167-6423(82)90012-0.

[14] A. Chakrabarti, G. Cormode, R. Kondapally, and A. McGregor. Information cost tradeoffs for augmented index and streaming language recognition. *SIAM Journal on Computing*, 42(1):61–83, 2013. 10.1137/100816481.