hexagon:2611.00311v1

NONSTABILIZATION OF THE PREDICTIVE BOOLEAN LOOKAHEAD HIERARCHY

WILL HOWE, GRANT HUFFMAN, JENSEN KOHLMEYER, AND LIAM KRUER

Abstract. We study the hierarchy of languages generated by predictive Boolean LL(k) grammars. The model uses a deterministic recursive-descent table, strongly non-left-recursive rules, and at least one positive conjunct per rule. For every integer $r \geq 1$, we construct a language over a fixed eight-letter alphabet that has an ordinary LL($r + r2^r + 2$) grammar and establish that it has no predictive Boolean LL(r) grammar. The construction asks for a specified token in a uniquely decodable word. The lower bound represents parsing on a long unary prefix by finitely many forward-moving endpoint streams and a finite Boolean recurrence. Random binary blocks followed by homogeneous blocks separate streams with a persistent timing error from streams having an exact finite-state potential. A bounded-lookahead argument hides a token-count change from each remaining stream. An encoding of the resulting independent queries is then shorter than their arbitrary binary payload. This gives nonstabilization under the stated predictive convention. The conclusion is for the stated parser model, including positive-conjunct and strong non-left-recursion requirements. Path-dependent tables satisfying these requirements give the same language families.

1. Introduction

Boolean grammars extend context-free grammars by allowing conjunction and negation in grammar rules. Their language semantics is independent of any particular parser. Okhotin's recursive-descent construction identifies a predictive subclass for which a bounded initial segment of the unread input selects the rule to be used [3].

Problem 8 in Okhotin's list asks whether the corresponding hierarchy eventually stabilizes [1, 4]. The question concerns language families. A language lower bound must exclude every grammar for that language with the smaller lookahead. The lower bound below quantifies over every predictive Boolean grammar with that lookahead and the premises in Section 2.

Let $\mathsf{BoolLL}(k)$ denote the family defined in Section 2, and let $\mathsf{LL}(k)$ denote the ordinary context-free LL(k) family. Throughout the paper, lookahead counts raw terminal symbols. A short unread suffix is distinguished from a longer suffix with the same initial symbols.

Theorem 1.1. *For every integer $r \geq 1$ there is a language L_r over the alphabet $\{a, b, c, d, e, p, x, y\}$ such that*

$$L_r \in \mathsf{LL}(r + r2^r + 2) \setminus \mathsf{BoolLL}(r). \tag{1.1}$$

Consequently, there is no integer $k_0 \geq 1$ for which $\mathsf{BoolLL}(k) = \mathsf{BoolLL}(k_0)$ for every $k \geq k_0$.

The proof uses the operational structure of the predictive parser. A long unary prefix gives each successful long return a single source endpoint. The other conjuncts test that endpoint, while fixed suffix procedures move it forward. This produces a finite collection of numerical streams. Their number may depend arbitrarily on the hypothetical grammar.

We construct enough independent changes of token count to hide one mode coordinate from each stream. At common query times, the parser would then recover an arbitrary array of binary strings

Date: November 7, 2026.

The authors acknowledge assistance from OpenAI Codex in mathematical exploration, checking, and manuscript preparation.

from fewer bits than the array contains. The argument permits the very large input lengths needed for successive calibrations by storing only equality information about remote endpoints.

For linear conjunctive grammars, Olkhovsky and Okhotin proved a collapse to lookahead one [6]. The present witness has a nonlinear production, and its lower bound addresses the full predictive Boolean family. Theorem 1.1 proves eventual nonstabilization for the explicit predictive convention used here. The relation between this convention and the short problem statement is discussed in Section 9.

2. The predictive model

A Boolean grammar is a tuple $G = (\Sigma, V, P, S)$ with finite terminal alphabet Σ, finite nonterminal set V, start symbol S, and rules

$$A \longrightarrow \alpha_1 \,\&\, \cdots \,\&\, \alpha_m \,\&\, \neg\beta_1 \,\&\, \cdots \,\&\, \neg\beta_t, \qquad m \geq 1, \quad t \geq 0. \tag{2.1}$$

Every α_i and β_j is a finite word in $\Sigma \cup V$. The language of a rule is the intersection of its positive-conjunct languages and the complements of its negative-conjunct languages. The language of a nonterminal is the union of its rule languages. We use the strongly unique language-equation semantics. The hypotheses below are the hypotheses of the predictive recursive-descent construction [2, 3].

Write G^+ for the positive transform obtained by deleting all negative conjuncts. A *signed occurrence* of B in a rule of A is an occurrence in either a positive or a negative conjunct. If that conjunct has body $\gamma B \delta$, its prefix is γ and its suffix is δ. The *zero-consumption graph* has an edge $A \to B$ for each such occurrence with $\varepsilon \in L_{G^+}(\gamma)$.

Definition 2.1. A grammar is *strongly non-left-recursive* if its zero-consumption graph is acyclic. A syntactic follower of an occurrence is obtained by concatenating the suffixes on a signed path from the start symbol to that occurrence, in reverse path order.

For a word w, let $\mathrm{First}_k(w)$ be its first k symbols, or all of w when $|w| < k$. A *predictive table* is a partial function

$$T \colon V \times \Sigma^{\leq k} \longrightarrow P.$$

It is valid if, for every rule $A \to \varphi$, every signed occurrence of A with syntactic follower η, and every $u \in L_G(\varphi) L_G(\eta)$, the entry $T(A, \mathrm{First}_k(u))$ is that rule.

The parser for a nonterminal selects this entry from the unread suffix. An absent entry fails. A selected rule executes its first positive conjunct and records the returned endpoint. Each remaining positive conjunct must return that same endpoint from the same starting position. Each negative conjunct must fail or return a different endpoint. The call succeeds exactly when these tests hold, and then returns the recorded endpoint. A sequence is parsed by executing its factors from left to right. The start call accepts exactly when it succeeds at the end of the input.

Strong non-left-recursion guarantees termination on finite inputs, including inputs outside an occurrence's syntactic follower domain. A successful operational return consumes a word in the corresponding language of G^+. On legal followers, the table construction gives correct recognition of the Boolean language. These facts follow from the recursive-descent soundness and termination argument [2, Sections 4–5]. We will also use their elementary operational content. Any chain of calls whose preceding successful factors consume zero symbols follows the acyclic zero-consumption graph. Every other edge advances the input pointer.

Definition 2.2. The family $\mathsf{BoolLL}(k)$ consists of languages of strongly non-left-recursive Boolean grammars of the form (2.1) admitting a valid predictive table with lookahead k.

The positive-conjunct premise is part of this definition. A purely negative rule has no positive endpoint for the stated parser to return. Appendix A treats the path-dependent version of the table condition in the original problem statement.

3. The token language

Fix $r \geq 1$ and put

$$J = r + r2^r, \qquad k = J + 2.$$

Define

$$C_J = \{a, b, e, x, y, b^{J+1}d\} \ \cup \ \{ab^j dve : 0 \leq j \leq J, \ v \in \{x, y\}^*\}. \tag{3.1}$$

Members of C_J are called tokens. A token in the second set is a long a-token. Define

$$L_r = \{p^n cw : n \geq 1, \ w \in C_J^*, \ \text{the } n\text{th token of } w \text{ is } a\}. \tag{3.2}$$

In particular, the queried token must exist.

Lemma 3.1. *Every word in C_J^* has a unique tokenization. At the beginning of a token, k symbols of raw lookahead determine its kind. There is an ordinary* LL(k) *grammar for L_r.*

Proof. If the remaining word begins with $ab^j d$ for $0 \leq j \leq J$, then the first token must be a long a-token. A singleton a would leave at most J consecutive b's before the first d, which no subsequent token could absorb. Conversely, a long a-token has exactly such a prefix. Its $\{x, y\}^* e$ tail is deterministic.

At a b-position, the token $b^{J+1}d$ is forced precisely when that prefix is present. Taking an initial singleton b would again leave too few b's to absorb the d. All other starts use a singleton. Induction over the successive tokens proves uniqueness. Each decision uses at most $J + 2$ raw symbols and remains valid with any follower in C_J^*.

More explicitly, let T generate one token by the rules

$$T \to a \mid b \mid e \mid x \mid y \mid b^{J+1}d \mid ab^j dVe \quad (0 \leq j \leq J),$$
$$V \to xV \mid yV \mid \varepsilon.$$

The alternatives of V are distinguished by the next symbol, since its follower is e. Let R be a deterministic right-linear grammar for the regular language C_J^*. Use

$$S \to pBaR, \qquad B \to pBT \mid c. \tag{3.3}$$

The B-recursion matches the first $n - 1$ tokens after c, and the literal a matches the nth. A literal a cannot instead be the first symbol of a long token, since its remainder would not belong to C_J^*. The alternatives of B have different first symbols. The described grammar for T is LL(k) on its followers, and R is needed only with the end of input as follower. Thus (3.3) has the required predictive grammar. □

We henceforth assume, for a contradiction, that a grammar G in the model of Section 2 recognizes L_r with lookahead r. All constants may depend on G. The alphabet, r, and the language in (3.2) remain fixed.

4. Endpoint locality and unary-prefix structure

For a nonterminal or fixed sequence α, write $\mathcal{P}_\alpha(w)$ for the length consumed by its actual parser on w, with value $\perp$ on failure. A successful length is always nonnegative. A suffix procedure therefore induces a partial extensive map on positions, meaning that an output position is at least the input position.

Lemma 4.1 (Exact endpoint locality)**.** *If* $\mathrm{First}_{\ell+r}(w) = \mathrm{First}_{\ell+r}(w')$*, then*

$$\mathcal{P}_\alpha(w) = \ell \quad \Longleftrightarrow \quad \mathcal{P}_\alpha(w') = \ell$$

for every fixed sequence α and every $\ell \geq 0$.

Proof. First treat nonterminals, by induction on ℓ and then by a topological ordering of the zero-consumption graph, with dependencies before dependents. Rule selection uses the same r-prefix on both inputs. A sequence returns exactly ℓ precisely when its factors return lengths $\ell_1, \dots, \ell_s \geq 0$ summing to ℓ. For a factor beginning at offset u, its required prefix has length at most $u+\ell_i+r \leq \ell+r$.

A child of length less than ℓ is covered by the first induction. A child of length ℓ begins after a zero-length prefix. If that prefix can succeed, it is nullable in G^+, so the relevant dependency decreases the topological rank. Otherwise that factorization cannot contribute on either input. This also treats $\ell = 0$.

The selected rule succeeds at length ℓ exactly when all positive sequences succeed at that length and no negative sequence succeeds at that length. These conditions are preserved by the induction. The sequence assertion follows from the same factorization criterion. Equality of truncated prefixes also preserves a short end-of-input window, so the argument includes finite suffixes of length less than $\ell + r$. □

A failing call can inspect an arbitrarily long prefix. In particular, a negative branch may run far beyond the eventual positive endpoint. Returning at a specified endpoint is local in the stated sense.

4.1. **The fixed positive skeleton.** Fix a terminal u and consider inputs $u^n w$. On the infinite stream u^ω, follow the first positive conjunct at every call and omit all other tests. Each resulting skeleton has one of three types. It returns after a fixed finite length, fails after a finite computation, or continues indefinitely. There is a constant n_0 such that every finite skeleton has completed before the last r symbols of u^n whenever $n \geq n_0$.

An actual successful call has the same first-positive execution as its skeleton. Consequently, for $n \geq n_0$ a finite returning type can only return its fixed short length. A finite failing type cannot succeed. An indefinitely continuing type can succeed only with an endpoint within the last $r-1$ symbols of u^n or within w.

Measure a long endpoint relative to the beginning of w. It then lies in

$$\{-r+1, -r+2, \dots, |w|\}. \tag{4.1}$$

For each nonterminal A, let $F_A(n; w)$ record failure, a short successful return, or this long endpoint.

Lemma 4.2 (Finite source representation)**.** *For the fixed grammar and unary terminal u, there are integers $H \geq 1$, $P \geq 0$, and $D \geq 0$ with the following properties.*

1. *Long successful returns are obtained from P potential endpoint streams. Each stream, on its residue class of outer indices n, iterates one fixed suffix procedure Γ_i. One iteration increases n by H.*
2. *Each stream is nondecreasing while it moves. A zero step is stationary thereafter, and failure of the potential procedure is permanent. Actual success agrees with the potential endpoint whenever the relevant guards hold.*
3. *All success flags satisfy a deterministic finite Boolean recurrence. At outer index n, it uses a fixed finite list of suffix procedures evaluated at potential stream positions with indices in a window of width D about n.*
4. *Apart from these procedure evaluations, the numerical tests are equalities of endpoints and comparison with the final endpoint. The recurrence creates no procedure address by combining two endpoints.*

The constants and procedure lists are independent of w.

Proof. For a signed sequence in a selected rule, traverse the factors until the first indefinitely continuing skeleton. Every earlier successful factor has a fixed short length. Their total is a grammar-bounded delay δ. If there is a finite failure first, the sequence fails. If there is no long factor, its only possible successful length is fixed. Otherwise its endpoint is

$$h\big(F_B(n-\delta; w)\big),$$

where B is the first long factor and h is the actual parser for the remaining fixed sequence. This is a partial extensive map independent of n. Earlier short success flags and all signed tests remain as Boolean conditions.

For the first positive sequence of a long nonterminal A, this gives one designated source $B(A)$, one delay δ_A, and one suffix map h_A. Inaccessible and permanently failing sources can be removed. Every remaining component of this finite functional graph contains a cycle. A cycle of total delay zero would be a cycle in the original zero-consumption graph, so every cycle has positive total delay.

Choose H to be a common multiple of all cycle delays, taking $H = 1$ when there are no cycles. Following a cycle until the outer index has decreased by H expresses its endpoint as one fixed composition of suffix procedures applied to its earlier endpoint. For every cycle node and every outer residue modulo H, choose a fixed sufficiently large initial index. Seed the potential stream by the actual endpoint there, and iterate that composition in the forward direction. An actual success always agrees with its potential stream. An actual guard failure can suppress a value but cannot change the potential trajectory or introduce a new endpoint source.

Unroll the bounded paths from off-cycle nodes to their cycles. Compose their suffix procedures with the fixed procedures already needed for the signed sequences. This produces a finite list of maps evaluated only at cycle positions at finitely shifted times. The identity map may be included in the list.

Zero-delay conditions are evaluated in the topological order of the zero-consumption graph. All other delays are bounded. Positive conjuncts compare returned endpoints for equality, and negative conjuncts test inequality or failure. Short-long comparisons have a fixed result beyond an enlarged threshold. The resulting finite recurrence has the claimed form. □

Lemma 4.3 (Uniform leading-block periodicity)**.** *There are constants $d, d_0 \geq 0$ and $q \geq 1$ such that the complete operational endpoint vector on $u^m w$ is periodic in m with period q whenever $m \geq d|w| + d_0$. Long endpoints are measured relative to w, and short endpoints by their consumed lengths. The assertion also holds for any fixed finite list of suffix procedures.*

Proof. Use Lemma 4.2. On a fixed cycle residue, actual successful long endpoints are nondecreasing in the finite set (4.1). If an actual cycle value fails, the cycle formula forces every later value on that residue to fail. There are therefore $O(|w| + 1)$ changes among all cycle residues, including their transitions to failure.

During an interval with no such change, off-cycle potential values are periodic with period H after a fixed delay margin. They are fixed functions of the cycle values. The remaining state consists of finitely many success bits in a bounded memory window and the current residue. Its number of possibilities is bounded by a grammar-dependent constant M_0. An interval without a cycle change longer than the delay margin plus $M_0 + 1$ contains two identical full normalized states. The deterministic recurrence then repeats forever. Thus either periodicity has already begun, or another cycle change occurs within a bounded number of steps. Since the number of changes is $O(|w| + 1)$, the transient is linear in $|w|$.

The eventual period is at most M_0. Taking a common multiple of the integers from 1 to M_0 gives a period independent of w. A fixed suffix procedure is handled by adding a new one-rule wrapper, with no incoming dependencies, or directly by the sequence formula in the preceding proof. A finite list changes only the constants. □

We apply Lemma 4.2 to the unary terminal p. Positions will be measured from the beginning of the token word after the separator c. This changes the fixed offsets by one and has no effect on any conclusion. The number of streams is denoted by P. If $P = 0$, sufficiently long accepted words would have no long start endpoint, already contradicting (3.2). Hence $P \geq 1$.

5. Calibration of endpoint streams

A potential stream iterates a fixed actual suffix procedure Γ. We now examine its positive steps on an infinite binary input. This examination concerns the potential trajectory. Actual signed guards will be retained separately in the final Boolean simulation.

5.1. **A finite-state additive process.** For a fixed infinite word, a finite return of length ℓ means a return forced by its prefix of length $\ell + r$. Such a return is well-defined by Lemma 4.1. A zero return is stationary. An input admitting no finite return is sent to a cemetery outcome.

Lemma 5.1. *On independent fair binary letters, the successive r-windows at positive cutoffs, together with the consumed lengths, form a finite-state Markov additive process, with possible absorption at failure or stationarity. Its state space is a subset of $\{a, b\}^r$.*

Proof. At a current cutoff j, the known window is the next r symbols. The event that the next length is ℓ is determined by the input through position $j + \ell + r$. Consequently the new cutoff plus its window is a stopping boundary for revealing letters. Bits beyond that boundary are fresh and independent. Conditional on the current window, the distribution of the next window and length is therefore independent of the earlier trajectory. Overlapping windows when $\ell < r$ simply retain their already known letters. Zero returns and the absence of any finite return give the stated absorbing outcomes. □

Lemma 5.2 (Drift or an exact potential)**.** *Fix the positive integer H from Lemma 4.2. In a closed irreducible moving class of the process in Lemma 5.1, exactly one of the following applicable descriptions holds.*

(1) The mean progress per step is $\mu \neq H$, possibly $\mu = \infty$. If T_N is the first step crossing raw position N, then $T_N/N \to 1/\mu$, with $1/\infty = 0$.

(2) The mean is H, and for every $0 < \eta < 1/8$,

$$\Pr(|HT_N - N| \leq N^{2\eta}) \longrightarrow 0. \tag{5.1}$$

(3) There is a function ϕ on the class such that every possible step from q to q' has positive finite length

$$\ell = H + \phi(q') - \phi(q). \tag{5.2}$$

This formula and positive finite returns hold on every binary continuation from the class.

For a fixed procedure, there is at most one class of the third kind, and it contains the window b^r.

Proof. Successive returns to a fixed state form independent excursions. Their number of steps has finite mean. The strong law for excursion rewards gives a limiting mean raw progress μ per step. Truncating the nonnegative rewards proves divergence of the average when their mean is infinite. Positivity then gives the assertion about crossing times in the first case.

Suppose $\mu = H$. Regard different possible lengths on one transition as different edges. If two closed walks have different average lengths, repeat them to obtain two return paths having the same number h of steps but distinct total raw lengths. Each path has positive probability. Sample in blocks of h steps in an appropriate residue of the chain's period. The resulting finite chain is irreducible on its cyclic component. With probability tending to one, a positive fraction of the first t blocks are one of the two selected return paths.

Condition on the coarser trajectory which identifies the union of those two paths, and on every other block. Their remaining choices are independent Bernoulli variables with a fixed parameter in $(0, 1)$. They have the same initial state, final state, and duration, so choosing one does not change the distribution of the subsequent coarser path. The raw position is a fixed term plus a nonzero integer multiple of their sum. The largest atom of that sum is $O(t^{-1/2})$, by the binomial coefficient

estimate. Thus the position at time t has vanishing probability of lying in any interval of length $o(\sqrt{t})$.

To pass to crossing times, take $t = \lfloor N/H \rfloor$ and a window of $O(N^{2\eta})$ steps about t. The expected total progress in this window is $O(N^{2\eta})$, uniformly over its starting state. By Markov's inequality it exceeds $N^{3\eta}$ with probability tending to zero. A crossing within the window would therefore place the position at time t within $O(N^{3\eta})$ of N, except on that exceptional event. Since $3\eta < 1/2$, the preceding anti-concentration bound proves (5.1).

If all closed walks have the same average, their average is μ. The sum of $\ell - \mu$ on each closed walk is zero. Path integration therefore defines a potential with $\ell = \mu + \phi(q') - \phi(q)$. For $\mu = H$ this is (5.2). The finite potential range bounds all step lengths. If some binary continuation had no positive return by this bound, locality would give a positive-probability finite cylinder with the same property, contradicting closed almost-sure survival. Hence the formula is valid on every continuation.

Continuing with b forever eventually places every positive moving trajectory in state b^r. Closedness keeps this state in the class. Two different closed classes are disjoint, so there is at most one exact class for the procedure. □

5.2. **Passing from random input to a homogeneous block.** We will use a block Xb^R, where X is binary of length N and $R = N^2$. The next lemma prevents an altered return near the join from having an uncontrolled intermediate length.

Lemma 5.3 (Join bound)**.** *Let an actual successful fixed-sequence call start at position $j \le N$ of an input whose interval $[N, N + R)$ consists of b's. There is a grammar-dependent constant C such that its endpoint either lies within r positions of $N + R$ or later, or lies at most $C(N - j + 1)$ positions beyond N.*

Proof. Keep the successful first-positive parse forest of the call. At the first logical crossing of N, its pending stack consists of siblings along an active ancestor path and a fixed number of remaining top-level factors. Every path edge with a positive preceding consumption advances by at least one symbol, so there are at most $N - j$ such edges before the join. Consecutive zero-prefix edges follow the acyclic zero-consumption graph and have bounded length. The pending stack therefore has $O(N - j + 1)$ entries.

If the endpoint is at most $N + R - r$, each remaining entry of this successful forest finishes on homogeneous input with its full lookahead still in the b-block. Its first-positive computation on b^ω must then have the same finite length. There are finitely many types of pending factors, and their finite homogeneous return lengths have a fixed maximum. Summing these lengths proves the bound. The fixed number of factors in the original sequence changes only C. □

For finite mean progress, the last random cutoff before a specified raw position has tight undershoot. Indeed, for $A \ge 1$ the chance of an undershoot exceeding A is bounded by

$$\sum_{h \ge A} \max_q \Pr_q(\ell > h). \tag{5.3}$$

Every raw cutoff position is visited at most once, and the Markov property applies there. There are finitely many states and their lengths have finite means, so the right side tends to zero.

Lemma 5.4 (Uniform calibration)**.** *Fix finitely many stream procedures and finitely many possible finite preludes to a new binary block. Fix an entering clock discrepancy bound B, a bound L on all later raw input, and a constant $d_* > 0$. One can choose arbitrarily large N and a common binary word X of length N such that, after Xb^{N^2}, every stream and prelude has one of the following descriptions, uniformly in the later input.*

(1) It is dead or stationary by all relevant final query times.

(2) It has entered its exact class from Lemma 5.2. Its clock discrepancy changes by a bounded amount independent of N. Its deep homogeneous cutoffs are spaced by H, and their phase is determined before the later input.

(3) At the final query times it is still deep in this homogeneous block, at an affine position on each outer residue. Its unread homogeneous length exceeds $d_*(L+1)$.

The final queries may range over a prescribed interval after the nominal end of the block, provided its total length and its clock offset are bounded by $L + B$.

Proof. Couple all finite words $Xb^{N^2}v$ to the same infinite independent binary reference with prefix X. Every reference cutoff at most $N - r$ is reproduced on all these words by Lemma 4.1. A first changed successful return must end past $N - r$. A reference zero step before that position remains stationary. Conditional on eventual absorption, its time and last finite position are tight. Conditional on survival, the entry into one of the finitely many closed moving classes is also tight. Choose fixed bounds for these quantities with an arbitrarily small exceptional probability.

In a finite-mean class, (5.3) places the last cutoff before $N-r$ within a fixed distance of it, outside an arbitrarily small exceptional event. Only finitely many binary fragments can lie between that cutoff and N. On each fragment followed by b^ω, a finite homogeneous return has a fixed length. Locality and Lemma 5.3 show that the altered trajectory either enters deep homogeneous input at position $N + O(1)$ or jumps near its far end. There are at most $r - 1$ additional positive starts immediately before the join.

In the infinite-mean and absorbed-reference cases, Lemma 5.3 gives entry no farther than $O(N+1)$ into the homogeneous block, or a jump to its far end. The entry clock is $o(N)$ in the infinite-mean case and bounded on the retained absorbed event. These bounds also apply to an initial seed procedure with a fixed prefix.

On deep b input, the fixed procedure has one homogeneous skeleton. It either has one finite positive step length h, returns zero, fails, or can return on a finite word only near the end of the block. A jump near the end occurs at clock $O(N)$ and is far ahead of the nominal clock $N + N^2$. There are at most $L + O(1)$ subsequent strict position increases before failure or stationarity.

For a finite positive homogeneous step, $h > H$ also makes the stream far ahead, while $h < H$ leaves a positive fraction of the N^2 block unread at the final queries. If $h = H$, the discrepancy at the random join persists. Wrong mean creates an order-N discrepancy. Equation (5.1) makes a mean-H nonexact class miss a window of width $N^{2\eta}$ with probability tending to one. An absorbed or infinite-mean reference is ahead by order N on the retained event. Choose $N^{2\eta}$ much larger than any required multiple of $L + B$.

The remaining case is an exact class. All its binary steps are bounded and obey (5.2), including the continuation into the b-block. The bounded entrance data and finite potential range bound its discrepancy independently of N. Its homogeneous phase is known from the common prefix. This gives the second description.

All statements about the finite continuation followed from locality or the early-return versus far-jump dichotomy, and hence are uniform in v. Choose the exceptional probability smaller than the reciprocal of the finite number of procedure-prelude pairs. A union bound gives one reference prefix X that works for all pairs. Increasing N gives the required unread length and query margins. □

6. A hidden change of token count

We next use the bound on the number of possible lookahead windows. It is independent of the range of the potential in Lemma 5.2.

Lemma 6.1 (A common cutoff)**.** *Suppose a stream is in an exact class. After a fixed common prefix, there are two binary continuations differing at one position s, one containing a there and*

one containing b, for which the stream reaches the same cutoff j_* at the same outer time. Moreover,

$$1 \le j_* - s \le r2^r + 1.$$

A marker may be placed at position $j_* + r$, followed by any common suffix, without changing this common cutoff.

Proof. Fix the seed and write the cutoff after t steps as

$$j_t = Ht + c_0 + \phi(q_t). \tag{6.1}$$

Take the continuation after the seed to consist entirely of b's, and choose t large enough that the positions considered below are past the seed. Then $q_t = b^r$ and $j_* = Ht + c_0 + \phi(b^r)$.

For each state q of the exact class, the candidate endpoint $Ht + c_0 + \phi(q)$ has an r-symbol window. There are at most 2^r states, so these windows cover at most $r2^r$ raw positions. Among the $r2^r + 1$ positions immediately preceding j_*, choose s outside all candidate windows. Change only position s from b to a.

The changed continuation still has positive exact-class steps by Lemma 5.2, and still satisfies (6.1) with the same c_0. Every candidate endpoint has unchanged window b^r. Its state must therefore be b^r, giving the same endpoint j_* at step t.

All earlier cutoffs are at most j_*. Inserting a marker at $j_* + r$ changes none of their consumed prefixes and following r-windows. Lemma 4.1, applied successively, preserves every cutoff up to j_*. The remaining raw suffix is the same in the two inputs, so the potential trajectories from this cutoff onward are identical. □

Lemma 6.2 (Token-count change)**.** *In Lemma 6.1, place d at $j_* + r$ and then append $x^v e$. By choosing $v \ge 0$, the b-signal version can have any prescribed positive number W more tokens than the a-signal version. Both versions have the same raw length.*

Proof. Put $B = j_* - s$. The number of b's between the signal and d is

$$j = B + r - 1 \le r2^r + r = J.$$

Choose the common leading homogeneous segment long enough to contain at least $J + 1$ consecutive b's before d. In the b-signal version, the last $J + 1$ of these and the marker form the special token $b^{J+1}d$. The following $x^v e$ contributes $v + 1$ singleton tokens. In the a-signal version, the signal and $b^j dx^v e$ form one long token.

If the common binary segment before d has length ℓ, these portions contribute respectively

$$\ell - J + v + 1 \quad \text{and} \quad \ell - j$$

tokens. Their difference is $v + 1 - (J - j)$. Taking $v = W + J - j - 1$ makes the difference W and gives $v \ge 0$. □

7. Simultaneous modes

We construct a finite family of words on which each stream loses one independently chosen coordinate. The family depends on the alleged grammar G, while its words are all words in the fixed code language (3.1).

Choose a power of two $M = 2^m$ and put

$$Q = M^2, \qquad \Lambda = M^P Q, \qquad \theta = (\theta_1, \dots, \theta_P) \in \{0, \dots, M-1\}^P. \tag{7.1}$$

The payload $z \in \{a, b\}^\Lambda$ is arbitrary. It is partitioned into M^P successive blocks of length Q. Define

$$I(\theta) = \sum_{i=1}^{P} \theta_i M^{i-1}.$$

Stream i owns a consecutive group of m signal positions, one for each binary digit of θ_i. The signal for digit b, where $0 \le b < m$, is given token-count weight

$$W_{i,b} = QM^{i-1}2^b. \tag{7.2}$$

Signal a represents a digit one and reduces the token count by $W_{i,b}$ relative to signal b.

7.1. **Choosing the raw blocks.** Before each signal place a calibration block Xb^{N^2} as in Lemma 5.4. Beyond its nominal end, reserve a homogeneous margin and place the signal and marker from Lemma 6.1. The margin is chosen large enough that all candidate windows lie after the preceding fixed cutoff and that the special token containing d lies wholly in this margin and its trap. This is possible because G and its potential ranges are fixed.

For a stream in an exact class, take the first suitable cutoff in the margin. Its position depends on its phase modulo H and varies by at most H. Compensating singleton b's after e make the total raw trap length and its all-b token count independent of that phase. Use a predetermined placement when the stream is already excluded or stationary. Lemma 6.2 assigns the required weight in (7.2) in either placement.

All trap lengths and margins are fixed before any calibration sizes are chosen. The possible local inputs between a safe cutoff at the end of one calibration and the next binary block form a finite set. Indeed, an exact stream has a cutoff within $H + r$ positions of the nominal homogeneous end, with lookahead entirely before the next signal. The intervening word consists of a bounded margin, one fixed-length trap, and its compensating padding. The initial seed procedures have a fixed p-prefix and the separator c, and are included among the finitely many initial cases.

Choose the entrance bounds in Lemma 5.4 jointly over these finite sets. Let B_0 bound all accumulated discrepancies of streams that remain exact, all fixed source delays, and all differences between raw length and token count in the traps. The constant B_0 may depend on G and M but is chosen independently of the later calibration lengths.

Append a final homogeneous padding b^S before z, where S dominates B_0, the last trap length, Q, and the fixed delay margin. Now choose the calibration sizes from right to left. At each stage choose N so large that its separation scale $N^{2\eta}$ dominates B_0 and the entire raw length of all later stages, traps, padding, and payload, multiplied by every grammar-dependent constant used in Lemma 4.3. Apply Lemma 5.4 to the already fixed finite prelude set. Fix a common successful binary reference prefix X for that stage.

These choices have no circular dependence. The finite prelude words use only the bounded pieces after safe cutoffs, so their lengths do not contain earlier or later nominal calibration lengths. The separation estimate is uniform in the remaining suffix. Each stage can therefore be fixed before any payload bits are selected.

Lemma 7.1 (Persistence of exclusion)**.** *A stream excluded by a calibration cannot supply a moving endpoint near the payload at the final query times. It is dead or stationary, or it remains deep in the homogeneous block of that calibration. In the last case its position is affine on each outer residue, and its unread homogeneous length is beyond the linear threshold of Lemma 4.3 for every procedure subsequently used.*

Proof. An ahead stream has more available outer steps than the number of remaining raw positions, after division by the fixed step spacing H and allowance for all delay margins. Every positive step advances at least one position. It must therefore fail or become stationary before the queries. Once a fixed procedure returns zero at a fixed position, it returns zero there on every further iteration.

A lagging stream advances by the fixed homogeneous step h on its outer residue. The discrepancy produced at its excluding stage is larger than all later raw lengths, clock corrections and query widths by the right-to-left choice of scales. Thus its unread homogeneous length stays above any prescribed grammar multiple of the remaining suffix length throughout the expanded query interval.

Choose that multiple to include the constants of Lemma 4.3 for the complete finite procedure list. □

7.2. **The number of layouts.** A *layout* specifies the raw placement of every signal and marker. Signal values are not part of the layout.

Lemma 7.2. *The number of possible layouts is at most*

$$K = (H+1)^P. \tag{7.3}$$

The layout of a mode is independent of z. Moreover, for any stream i, two modes with equal coordinates other than i and equal full layout have identical potential trajectories, at the same outer times, after their last i-owned signal, whenever their informative trajectories reach all those signals.

Proof. At the first signal of group i, its stream has one of H incoming homogeneous phases, or it uses the excluded fallback. Fix this choice. The first marker placement is now determined. At the common cutoff from Lemma 6.1, the remaining raw word is identical for the two values of the signal. The local process in the next fixed calibration, its classification, and its next homogeneous phase are consequently independent of the signal just hidden. These are properties of the local suffix and the fixed procedure, independent of the absolute outer time. If the stream becomes excluded, the remaining placements in that group use the fixed fallback. Induction over its m signals shows that its whole group's layout is determined by its incoming phase or fallback.

Different earlier groups can change a later group's incoming phase. There are nevertheless at most $H+1$ choices per group, giving (7.3). The classification uses the fixed reference prefixes and uniform finite-tail estimates, so it is independent of the arbitrary payload.

For the stronger same-time assertion, fix all other mode coordinates and the complete layout. Before group i the raw inputs coincide. Endpoint locality gives equal seed positions and equal outer times up to a safe cutoff before its first signal. An informative stream cannot have skipped that cutoff in an earlier ahead jump, by Lemma 7.1. Lemma 6.1 gives equality of position and time through the first signal. Repeat this reasoning through all signals of the group. Thereafter the entire remaining raw suffix is common, since all other digits and placements agree. The potential trajectories are therefore identical at the common query times. Actual success flags can still differ and will be recorded separately. □

Let U_θ be the resulting prefix, including the final padding and excluding the payload. Its raw length is independent of θ. Its number of tokens is

$$\#\,\mathrm{tokens}(U_\theta) = K_0 - QI(\theta), \tag{7.4}$$

where K_0 is the count when every signal is b. The compensating pads make K_0 independent of the layout. Every word $U_\theta z$ belongs to C_J^*, and all payload letters are singleton tokens. At the common indices

$$n = K_0 + q, \qquad 1 \le q \le Q, \tag{7.5}$$

membership of $p^n c U_\theta z$ in L_r reports exactly whether the $(QI(\theta)+q)$th payload symbol is a.

8. A finite description of the query interval

The calibration blocks can be far longer than the payload. Encoding an unrestricted raw position would therefore give an inadequate bound. We now describe only the numerical information the Boolean recurrence actually uses.

Include the fixed delay margin of Lemma 4.2 in the query interval. Its width is $Q + O_G(1)$. Use the beginning of the payload as the origin for the following coordinates.

Lemma 8.1 (Per-mode summary)**.** *For each mode θ and payload z, the recurrence on the query interval can be simulated from the following data.*

(1) A per-mode summary of $O_G(\log \Lambda)$ bits.
(2) For each stream that needs near-payload information, a table of $O_G(Q \log \Lambda)$ bits containing its finite list of map outputs on the expanded query interval.

A near-payload table for stream i is identical for two modes with the same other coordinates and the same layout, whenever that table is needed for both modes.

Proof. By the final padding, any informative moving stream that survived all calibrations has passed all its own signal cutoffs before the queries. After its last safe cutoff, only a fixed-length trap lies before that padding. A positive trajectory has at most that many strict advances before entering or passing the padding. A zero step or a failure makes it stationary or dead. The choice of S allows all fixed source delays as well. Lemma 7.2 therefore gives the final sharing assertion for any near-payload data.

Consider the other data. A lagging stream excluded earlier remains at a position whose unread suffix has form $b^t v$, with $t > d(|v| + 1)$ for all needed maps. On each sufficiently refined outer residue, its source position is affine. Lemma 4.3 makes each map output either failure, a fixed short increment from that source, or a fixed endpoint relative to v. Thus the numerical outputs are constants or affine functions of outer time, with slopes in a finite grammar-dependent set. A stationary source gives fixed outputs or failure. An ahead source is already stationary or dead.

A surviving moving source in the final padding is treated similarly while it is farther than the linear periodicity threshold from the payload. If it approaches that threshold by bounded homogeneous steps during the query interval, its entire relevant bounded-step trajectory lies within $C\Lambda$ positions of the payload for a constant $C = C(G)$. A homogeneous procedure without a finite positive step can make one first jump to that region or later. This needs only one split time and a bounded number of extra descriptors. All later endpoints stay in the region, since the maps are extensive.

Record near-payload map outputs directly in this common interval of length $O_G(\Lambda)$, using $O_G(\log \Lambda)$ bits each. Describe all remaining outputs by a fixed finite family of constants and affine functions on outer residues, possibly before and after a split time. Their absolute intercepts need not be recorded. The decoder needs precisely the following comparisons.

(1) Two near values are compared by their recorded coordinates.
(2) A descriptor that can enter the near interval during the query window has starting value $O_G(\Lambda)$, because its slope is bounded and the window has width $O(Q)$. Record that value and compute its later values. A descriptor that never enters the near interval cannot equal a near value.
(3) For every pair of descriptors, record their equality pattern on the query window, including the finite time shifts used by the recurrence. Equal slopes give equality everywhere or nowhere on their common support. Unequal slopes permit equality at at most one time, which is recorded relative to the window.

Include the final endpoint as a descriptor. There are only grammar-boundedly many maps, residues, time shifts, supports and pairs. The descriptor labels, their equality information, possible split times, and the initial Boolean memory therefore require only $O_G(\log \Lambda)$ bits per mode.

Lemma 4.2 uses no arithmetic on remote values other than the stated equality tests, and generates no new query address from them. Its update can consequently be simulated from this data. Guard failures are handled by the mode's own Boolean flags, so no assumption that different modes have the same actual success flags has been made. □

Proof of Theorem 1.1. The upper bound is Lemma 3.1. For the lower bound, fix the alleged grammar G and the constants P, H, K above. We show that the query answers would give an injective encoding of all Λ-bit payloads into fewer than Λ bits.

Use every layout in the encoding. For each of the M^P modes record its summary from Lemma 8.1. For each layout, stream i, and tuple of the other $P - 1$ mode coordinates, store the near-payload

table once whenever one is needed. The sharing assertion makes one table sufficient for all relevant modes in that group. Use a fixed-size placeholder when no table is needed. Category flags in each mode's summary specify which tables its simulation uses.

There are at most KPM^{P-1} such tables. The total encoded length is therefore

$$O_G\Big(M^P \log \Lambda + KPM^{P-1}Q \log \Lambda\Big). \tag{8.1}$$

The grammar, chosen layouts, and calibration words are fixed for this encoding and do not depend on z. Hardwire them in the decoder. There is no effectiveness or time requirement on choosing these finite parameters.

The decoder runs the finite recurrence for every mode at the common indices (7.5). Equations (7.4) and (7.5) recover every one of the M^P payload blocks. Thus the encoding is injective on all 2^Λ payloads.

Since $Q = M^2$ and $\log \Lambda = (P+2)\log M$, dividing (8.1) by $\Lambda = M^P Q$ gives

$$O_G\left(\frac{\log M}{M^2} + \frac{KP \log M}{M}\right) \longrightarrow 0.$$

For sufficiently large M, a fixed-length padded version of the encoding is strictly shorter than Λ. The pigeonhole principle gives a contradiction.

Finally, ordinary LL(k) grammars belong to the Boolean family by using only positive single-conjunct rules. If the Boolean families stabilized at k_0, the language supplied by the theorem with $r = k_0$ would belong to $\mathsf{BoolLL}(k_0)$, contradicting the lower bound. This proves nonstabilization. □

9. Scope and relation to earlier results

The constructed language uses a fixed alphabet and is ordinary context-free. Its definition depends on r, while the large input parameters in the lower bound depend on the hypothetical competing grammar. This order of quantifiers is essential. The proof excludes every such grammar for one fixed L_r.

Okhotin's expressive-power work establishes unary regularity and restrictions on long homogeneous suffixes for Boolean predictive languages [5]. The present argument uses operational returns on a homogeneous prefix, followed by several independently controlled changes of token count. The finite Boolean recurrence allows guards to retain arbitrary earlier success information while constraining how new numerical endpoints are obtained.

The linear-conjunctive collapse of Olkhovsky and Okhotin [6] permits at most one nonterminal per conjunct. The production $B \to pBT$ in the witness uses two. The higher-level grammar can therefore process a sequence of variable-length tokens without lying in that restricted family.

The short definition accompanying the original problem states a pathwise rule-uniqueness condition. It does not repeat the positivity and non-left-recursion assumptions used by the recursive-descent construction. Our theorem concerns the model in Section 2. These premises cannot be discarded from our statement: a purely negative rule has no positive endpoint for the parser to return, and a non-consuming recursion can prevent termination despite rule uniqueness. Thus the theorem does not address arbitrary Boolean grammars satisfying only the short pathwise predicate. Appendix A shows that path-dependent and uniform tables give the same language families when the stated parser premises are retained.

Appendix A. Path-dependent and uniform tables

The original statement permits a rule uniqueness condition at each fixed signed path. For completeness, we give the finite annotation that converts it to one uniform table.

Proposition A.1. *Suppose a strongly non-left-recursive Boolean grammar of the form* (2.1) *satisfies the k-lookahead uniqueness condition at each fixed signed path from its start. It has an equivalent*

grammar with a uniform predictive k-table. The construction preserves the number of nonterminals per conjunct.

Proof. For a syntactic word η, put $F(\eta) = \mathrm{First}_k(L_G(\eta))$. For finite sets $X, Y \subseteq \Sigma^{\leq k}$, define

$$X \odot Y = \{\mathrm{First}_k(xy) : x \in X,\ y \in Y\}.$$

Truncation respects concatenation, so this operation gives the truncated follower language of a concatenated tail, including short end-of-input words.

Make a copy A_F of each old nonterminal for each possible finite set $F \subseteq \Sigma^{\leq k}$. The start symbol is $S_{\{\varepsilon\}}$. Copy every old rule for each A_F. In a signed occurrence of B with remaining local suffix δ, replace that occurrence by $B_{F(\delta)\odot F}$. Retain terminals and signs.

The candidate languages $L(A_F) = L_G(A)$ satisfy all copied equations. Projection of positive derivation trees shows that a nullable prefix in the new positive transform projects to a nullable prefix in the old one. Every new signed zero-consumption cycle would therefore project to an old such cycle. Strong non-left-recursion and the corresponding uniqueness of the equations give the asserted candidate languages as the actual languages of the copies.

Along every new signed path, the truncated language of its right tail is exactly the tag on the current nonterminal. This follows by induction from the start tag and the update $F(\delta) \odot F$. All occurrences of a fixed A_F thus have the same relevant follower-key set. If two of its copied rules required the same lookahead entry, their original rules would both require that entry at any one projected occurrence with this tail tag. This contradicts the assumed pathwise uniqueness. Unreachable copies impose no constraint. A uniform deterministic table follows.

The reverse implication is immediate, and each copied conjunct retains the old number of nonterminal occurrences. □

References

[1] A. Okhotin, *Nine open problems on conjunctive and Boolean grammars*, Bulletin of the European Association for Theoretical Computer Science **91** (2007), 96–119. `https://eatcs.org/images/bulletin/beatcs91.pdf`.

[2] A. Okhotin, *Recursive descent parsing for Boolean grammars*, Technical Report 2004–475, School of Computing, Queen's University, March 2004. `https://research.cs.queensu.ca/TechReports/Reports/2004-475.pdf`.

[3] A. Okhotin, *Recursive descent parsing for Boolean grammars*, Acta Informatica **44** (2007), 167–189. doi:10.1007/s00236-007-0045-0.

[4] A. Okhotin, *Nine open problems on conjunctive and Boolean grammars: an update*, manuscript, March 30, 2010. `https://citeseerx.ist.psu.edu/document?doi=d0e62d472c289c617866fde28ff6dd32758a73c7&repid=rep1&type=pdf`.

[5] A. Okhotin, *Expressive power of LL(k) Boolean grammars*, Theoretical Computer Science **412** (2011), no. 39, 5132–5155. doi:10.1016/j.tcs.2011.05.013.

[6] I. Olkhovsky and A. Okhotin, *On LL(k) linear conjunctive grammars*, 2021, arXiv:2112.08014 [cs.FL]. `https://arxiv.org/abs/2112.08014`.

Email address: `howew@purdue.edu`

Email address: `huffma97@purdue.edu`

Email address: `jenwin@purdue.edu`

Email address: `kruerl@purdue.edu`