THE STATE COMPLEXITY OF $Y = nX$ IN ZECKENDORF REPRESENTATION: EXACT VALUES, THE OEIS A372846 BOUND, AND THE REACHABLE STATES OF THE MORADI–RAMPERSAD–SHALLIT AUTOMATON

KOREA SUPERINTELLIGENCE LABS

Abstract. Let $a(n)$ be the number of states of the smallest deterministic finite automaton that reads two Zeckendorf representations in parallel and accepts exactly the pairs (x, y) with $[y] = n[x]$, the dead state not counted; this is sequence A372846 of the OEIS. Moradi, Rampersad and Shallit build an automaton $M_{n,c}$ with $36n^2$ states for the relation $Y = nX + c$, print $a(1), \ldots, a(10)$ as computed by `Walnut`, write that the quantity "is roughly $2n^2 + \alpha^2 n$, but we do not know how to prove this", and close the remark with "A better understanding of the reachable states of $M_{n,c}$ might help reduce the bound." The OEIS entry sharpens the guess to the conjecture $a(n) \le 2n^2 + \varphi^2 n + 1$. We prove $a(n)$ exactly for $1 \le n \le 16$ and $a(32) = 2132$ — to our knowledge the first proofs of any term — by an explicit automaton whose correctness is certified by integer arithmetic alone (two forward-closed half-planes in the plane of two consecutive differences replace the irrational estimate of the source) together with a Myhill–Nerode table for partial acceptors, in which liveness of the prefixes is an essential hypothesis. We show that the OEIS inequality, stated over the reals with $\sqrt{5}$, is equivalent to an integer inequality, and prove it at the nineteen values $n \in \{1, \ldots, 16, 32, 49, 50\}$; outside the formal development it is verified for every $n \le 601$, where the smallest slack is 0.652476 at $n = 14$ — not at $n = 32$, as the 48 published terms suggest — exactly two n have slack below 1, both proved here one state from the edge, and beyond $n \approx 40$ the slack grows like $0.10\,n$. We answer the closing sentence: the part of $M_{n,c}$ reachable from the initial state has exactly $6n^2 + 6n - 4$ states for every c, certified for $n \le 100$ and computed to $n = 601$, a factor 6 below the printed count. Past the 48 terms of the OEIS entry we certify $a(49) \le 4926$ and $a(50) \le 5128$, and two independent implementations agree on $a(49), \ldots, a(60)$ and continue to $a(601) = 723916$. All theorems are machine-checked in Lean 4.

1. Introduction

Zeckendorf's theorem [10, 11] writes every natural number uniquely as a sum of distinct, non-consecutive Fibonacci numbers F_i, $i \ge 2$, where $F_0 = 0$, $F_1 = F_2 = 1$ and $F_{i+2} = F_{i+1} + F_i$. Recording the coefficients most significant digit first gives a binary word with no factor 11, and a finite automaton can read such words and decide arithmetic relations between the numbers they denote; this is the Fibonacci numeration system, the setting of [7, 8] and the input format of the automated prover `Walnut` [9, 8].

Moradi, Rampersad and Shallit [1] study the state complexity of the basic relations in this system. For the relation $Y = nX + c$ with the multiplier $n \ge 1$ fixed and X, Y read in parallel on two tapes, they construct a deterministic automaton $M_{n,c}$ whose states are quadruples $[a', b', d, d']$ — the last letters read on the two tapes and the last two values of the difference $[y] - n[x]$, confined to the interval $I_n = [-n, 2n - 1]$ — and prove that it accepts the language

$$L_{n,c} \;=\; \{\, x \times y \in (\Sigma_2 \times \Sigma_2)^* : \; [y] = n[x] + c \,\}$$

[1, Theorems 7 and 11]. Counting the state set gives $2 \cdot 2 \cdot 3n \cdot 3n = 36n^2$ states, "which is $O(n^2)$" [1, proof of Theorem 7]. The remark that follows is the starting point of this paper; we quote it in full, omitting only its two citation markers [1, Remark 12]:

> We point out that the DFA $M_{n,c}$ in our construction is not, in general, minimal. However, we can construct our automaton in quadratic time and then apply a well-known minimization algorithm for DFAs by Hopcroft or Valmari to get the minimal automaton recognizing $Y = nX + c$ in $O(n^2 \log n)$ time.

> The number of states in the minimal DFA recognizing the language $L_{n,0}$ forms sequence A372846 in the On-Line Encyclopedia of Integer Sequences (OEIS). The values for $1 \le n \le 10$ (not counting the dead state), as computed by the free software `Walnut`, are $2, 10, 23, 40, 59, 85, 114, 146, 181, 224$. Numerical evidence suggests that this quantity is roughly $2n^2 + \alpha^2 n$, but we do not know how to prove this. A better understanding of the reachable states of $M_{n,c}$ might help reduce the bound.

Here $\alpha = (1 + \sqrt{5})/2$, which we write φ below. The OEIS entry [3], contributed by Shallit in July 2024, lists 48 terms, $a(48) = 4731$, and sharpens the guess into a conjecture, stated there in the entry's notation as `a(n) <= 2*n^2 + g^2*n + 1, where g = (1+sqrt(5))/2`, that is

$$a(n) \;\le\; 2n^2 + g^2\, n + 1, \qquad g = \tfrac{1+\sqrt{5}}{2}. \tag{1}$$

The entry carries no formula and no program, and the first author's thesis [2], which the entry links, repeats the remark above word for word (its Remark 46, on p. 67 of the PDF, printed page 57; only the citation markers differ) and adds nothing to it. As far as we have been able to determine — the arXiv listing; the citing-works indexes of OpenAlex, which lists no citing work, and Semantic Scholar, which lists one, the authors' base-k companion [5], whose current version announces [1] as forthcoming and does not mention the sequence; the OEIS entry; and the thesis, all read on 3 September 2026 — no term of the sequence has been proved, no verification range for (1) has been stated, and the reachable part of $M_{n,c}$ has not been counted.

The one-tape analogue is settled: Charlier, Rampersad, Rigo and Waxweiler [4] proved that exactly $2n^2$ states are necessary and sufficient for a DFA to accept the multiples of n in the Fibonacci numeration system, and [1] says of that result that "despite the close relationship of their result and ours, we see no direct way to get our result from theirs." The leading term $2n^2$ of (1) is thus the one-tape answer; what the two-tape problem adds is the linear term, and no reduction between the two is claimed here either. Section 4 of [1] concerns a different object, automata *with output* reading a single tape, which generate a sequence rather than accept a relation; the question studied here is on the acceptor side. The companion manuscript [6] treats the msd-first side of the shifted Fibonacci word, an output-automaton question, and the two developments share only their definitions of Zeckendorf words.

Results. Write $a(n)$ for the least number of states of a *partial* DFA (transitions may be undefined) recognising $L_{n,0}$; Remark 2.1 explains why this is the OEIS quantity.

(1) *Exact values.* $a(n)$ is determined for $1 \le n \le 16$ and for $n = 32$ (Theorem 5.1): the values agree with those printed in [1] and listed in [3], and each is proved from both sides, by an explicit automaton certified correct for every input and by a Myhill–Nerode table valid against every competing automaton.

(2) *The conjectured bound.* Inequality (1), a statement about a real number, is equivalent to an integer inequality (Proposition 6.1); it holds at $n \in \{1, \dots, 16, 32\}$ by the exact values and at $n = 49, 50$ by certified upper bounds (Theorem 6.2), and at $n = 14$ and $n = 32$ one additional state would refute it (Proposition 6.3). Outside the formal development, (1) is verified for every $n \le 601$; the minimum slack over that range is 0.652476 at $n = 14$, which the 48 published terms do not reveal as the global minimum, and the slack grows linearly beyond $n \approx 40$ (Remark 6.4). A least-squares fit of $a(n) - 2n^2$ on $300 \le n \le 601$ has slope 2.5036, against $\varphi^2 = 2.618\ldots$; so "roughly $2n^2 + \alpha^2 n$" over-states the linear coefficient, and (1) is safe but loose. This is a measurement, not a theorem.

(3) *The reachable part of $M_{n,c}$.* The transition function and the initial state of $M_{n,c}$ do not involve c, so its reachable part is the same for every c, and it has exactly $6n^2 + 6n - 4$ states, certified for every $1 \le n \le 100$ (Theorem 7.1) and computed for every $n \le 601$ (Remark 7.3). This answers the closing sentence of the remark quoted above: the printed count $36n^2$ over-counts by a factor of 6, uniformly in n.

(4) *Beyond the published terms.* $a(49) \le 4926$ and $a(50) \le 5128$ are certified (Theorem 8.1); two independent implementations agree that these are the exact values and continue the sequence to $a(60)$, and one of them to $a(601)$ (Remark 8.3).

Method. The correctness of an automaton for $L_{n,0}$ is proved in [1] through the estimate $-\beta^2 < [x0] - \alpha[x] < -\beta$ [1, Lemma 2(b)], $\beta = (1-\sqrt{5})/2$, which confines the running difference to I_n. We avoid the irrationals entirely by carrying *two* differences, $D = [y] - n[x]$ and $E = [y0] - n[x0]$: their joint update on a letter is linear with integer coefficients, and two half-planes in the (D, E)-plane are closed under every update and avoid $D = 0$ (Lemma 3.2). That is all a correctness proof needs, and it is checkable by a machine one state at a time. The lower bound is the Myhill–Nerode argument for *partial* acceptors (Theorem 4.1), where a prefix with no continuation in the language reaches no state at all, so pairwise separation of prefixes bounds the state count only if every prefix is live; Proposition 5.3(iii) shows the hypothesis cannot be dropped. Everything is then a finite check on explicit data: for each n a certificate carrying a labelled automaton, a homomorphism onto the claimed minimal automaton, and a separation table (Definition 4.2). Section 9 records what was delegated to compiled evaluation and what was not.

2. Preliminaries

2.1. **Zeckendorf words.** A *word* is a finite sequence $x = e_t \cdots e_2$ over $\Sigma_2 = \{0, 1\}$, read most significant digit first, and its *value* is $[x] = \sum_{2 \le i \le t} e_i F_i$; thus $[01001] = 6$. A word is *valid* if it contains no factor 11; leading zeros are allowed, and every natural number is the value of exactly one valid word without leading zeros. The conventions are those of [1, §2.2]. Two identities drive everything below: for every word x and digit a,

$$[xa] = [x0] + a, \qquad [xa0] = [x] + [x0] + 2a, \tag{2}$$

the second being [1, Lemma 2(a)], $[x00] = [x] + [x0]$, with the digit a carrying $F_3 = 2$.

2.2. **Pairs and the language.** Pairs of numbers are read on two tapes in parallel over the alphabet $\Sigma_2 \times \Sigma_2$, the shorter representation padded with leading zeros [1, §2.2]: a word $w = (a_1, b_1) \cdots (a_\ell, b_\ell)$ over $\Sigma_2 \times \Sigma_2$ has two *tracks* $x = a_1 \cdots a_\ell$ and $y = b_1 \cdots b_\ell$, and we write $w = x \times y$. It is *valid* when both tracks are. For $n \ge 1$,

$$L_{n,0} = \{\, x \times y : \ x \times y \text{ valid}, \ [y] = n[x] \,\}$$

is the language of [1, §3.2] at $c = 0$. In the formal development the pair language is `Lang n`, and its definition is exactly this: both tracks valid, and $[y] = n[x]$.

2.3. **Partial automata and $a(n)$.** A *partial DFA* on m states is a triple (δ, F, q_0) with $\delta : \{0, \dots, m-1\} \times (\Sigma_2 \times \Sigma_2) \rightharpoonup \{0, \dots, m-1\}$ a partial function, F a set of accepting states and q_0 the initial state. It *accepts* w if the run from q_0 on w is defined throughout and ends in F, and it *recognises* L if it accepts exactly the words of L. We set

$$a(n) = \min\{\, m : \text{ some partial DFA on } m \text{ states recognises } L_{n,0} \,\}.$$

In the development this is `SC (Lang n) m`, defined as the statement that m is the least element of the set `SizeSet (Lang n)` of sizes of recognising partial DFAs; so every theorem $a(n) = m$ below asserts both that an m-state automaton exists and that no smaller one does.

Remark 2.1. This is the OEIS quantity "not counting the dead state". Deleting the dead state of a complete DFA for $L_{n,0}$ leaves a partial DFA recognising $L_{n,0}$, and adjoining a sink to a partial DFA gives a complete one with one more state; the minimal complete DFA of $L_{n,0}$ does have a dead state, since a word containing 11 on a track has no extension in $L_{n,0}$. Hence the least size of a complete DFA is $a(n) + 1$, and $a(n)$ as defined here is the number in [1, Remark 12] and [3].

2.4. **The automaton of the source.** For $n \ge 1$ and $0 \le c < n$ (the standing assumption of [1, §3.2]), the automaton $M_{n,c}$ of [1, §3.2] has state set $Q = \{[a', b', d, d'] : \ d, d' \in I_n, \ a', b' \in \Sigma_2\}$ with $I_n = [-n, 2n-1]$ [1, Theorem 11], initial state $q_0 = [0, 0, 0, 0]$, accepting set $\{[a', b', d, d'] : d = c\}$, and transition

$$\delta([a', b', d, d'], [a, b]) = [a, b, g, d] \quad \text{if } g \in I_n, \text{ where } g = d + d' + b - na + b' - na', \text{ and } aa', bb' \ne 11,$$

undefined otherwise. The intended meaning is that after reading $xa' \times yb'$ the automaton is in the state $[a', b', D(xa', yb'), D(x, y)]$ with $D(x, y) = [y] - n[x]$, the transition rule being [1, Lemma 4], $D(xa'a, yb'b) = D(x, y) + D(xa', yb') + b - na + b' - na'$. Write $\mathrm{Reach}(M_{n,c})$ for the set of states reachable from q_0.

3. An integer-only correctness criterion

For a word $w = x \times y$ over $\Sigma_2 \times \Sigma_2$ and $n \geq 1$ put

$$D_n(w) = [y] - n[x], \qquad E_n(w) = [y0] - n[x0].$$

By (2), reading one more letter (a, b) updates the pair linearly:

$$D_n(w(a,b)) = E_n(w) + b - na, \qquad E_n(w(a,b)) = D_n(w) + E_n(w) + 2b - 2na. \tag{3}$$

Definition 3.1. The *dead region* is $\mathsf{Dead}_n = \{(d, e) \in \mathbb{Z}^2 : (2n \leq d \wedge 3n \leq e) \vee (d \leq -2 \wedge e \leq -3)\}$.

Lemma 3.2. *Let $n \geq 1$ and $(d, e) \in \mathsf{Dead}_n$. Then for all $a, b \in \{0, 1\}$, $(e + b - na,\ d + e + 2b - 2na) \in \mathsf{Dead}_n$, and $d \neq 0$. Consequently, if $(D_n(w), E_n(w)) \in \mathsf{Dead}_n$ then no extension wz lies in $L_{n,0}$.*

Proof. Three additions in each half-plane. If $2n \leq d$ and $3n \leq e$ then $e + b - na \geq 3n - n = 2n$ and $d + e + 2b - 2na \geq 2n + 3n - 2n = 3n$; if $d \leq -2$ and $e \leq -3$ then $e + b - na \leq -3 + 1 = -2$ and $d + e + 2b - 2na \leq -2 - 3 + 2 = -3$. In the first half-plane $d \geq 2n \geq 2$, in the second $d \leq -2$. The last sentence follows by induction along z using (3), since $wz \in L_{n,0}$ forces $D_n(wz) = 0$. (Formally, `Dead_step`, `Dead_ne_zero` and `not_lang_of_dead`.) □

The constants of the lower half-plane do not depend on n; this small observation is what keeps the whole apparatus inside $\mathbb{Z}$. The source's route instead confines D to I_n through the estimate $-\beta^2 < [x0] - \alpha[x] < -\beta$ [1, Lemma 2(b), Lemmas 8–10]; nothing below uses α, β or I_n.

Definition 3.3. A *labelling* of a partial DFA A on M states is a map $\lambda : q \mapsto (d_q, e_q, a'_q, b'_q) \in \mathbb{Z} \times \mathbb{Z} \times \Sigma_2 \times \Sigma_2$. Say a letter (a, b) is *blocked* at q if $a = a'_q = 1$ or $b = b'_q = 1$. The labelling is *admissible for n* if

- (L1) $\lambda(q_0) = (0, 0, 0, 0)$;
- (L2) q is accepting if and only if $d_q = 0$;
- (L3) $\delta(q, (a, b))$ is undefined whenever (a, b) is blocked at q;
- (L4) if (a, b) is not blocked at q and $\delta(q, (a, b)) = q'$ is defined then $\lambda(q') = (e_q + b - na,\ d_q + e_q + 2b - 2na,\ a,\ b)$, and if it is undefined then $(e_q + b - na,\ d_q + e_q + 2b - 2na) \in \mathsf{Dead}_n$.

Theorem 3.4. *Let $n \geq 1$. A partial DFA with an admissible labelling for n recognises $L_{n,0}$.*

Proof sketch. By induction on w, using (L3), (L4) and (3): either the run on w is defined, w is valid and the state reached carries the label $(D_n(w), E_n(w),$ last letters of $w)$, or w is invalid, or the run has stopped at a valid prefix whose true (D, E) is in Dead_n, and then so is that of w by Lemma 3.2. (L2) and Lemma 3.2 finish both directions. (Formally `run_lab` and `recognizes_of_labels`.) □

The minimal automaton cannot be labelled in this way, since one of its states merges several values of (D, E). Recognition is transported to it along the class map of the minimisation:

Theorem 3.5. *Let B be a partial DFA on M states and A one on m states, and let $h : \{0, \ldots, M-1\} \rightharpoonup \{0, \ldots, m-1\}$ be a partial map such that $h(q_0^B) = q_0^A$; for every state q and letter l, if $h(q)$ is undefined then $h(\delta_B(q, l))$ is undefined, and if $h(q) = i$ then $\delta_A(i, l) = h(\delta_B(q, l))$, both sides being undefined together; and q is non-accepting when $h(q)$ is undefined, while $i = h(q)$ is accepting in A exactly when q is in B. If B recognises L then so does A. (Formally `recognizes_of_hom`.)*

4. Myhill–Nerode for partial acceptors, and certificates

Theorem 4.1. *Let A be a partial DFA on m states recognising L, and let $w_1, \dots, w_k$ be words such that*

(i) *each w_i is* live*: $w_i z \in L$ for some z; and*
(ii) *the w_i are pairwise* separated*: for $i \neq j$ there is z with exactly one of $w_i z$, $w_j z$ in L.*

Then $k \leq m$. The same holds when (i) and (ii) are witnessed by a single finite list Z of suffixes: every w_i has some $z \in Z$ with $w_i z \in L$, and the bit vectors $([w_i z \in L])_{z \in Z}$ are pairwise distinct.

Proof. By (i) the run on each w_i is defined — a run that stopped could not be continued to an accepting one — and ends at some state q_i. If $q_i = q_j$ with $i \neq j$ then A treats $w_i z$ and $w_j z$ alike for every z, contradicting (ii). So $i \mapsto q_i$ is injective. The second form is the first with the witnesses made explicit. (Formally `length_le_of_sep` and `length_le_of_cert`; the second has both hypotheses as Boolean computations.) □

The liveness hypothesis (i) is where partial acceptors differ from automata with output, whose every valid input reaches a state; Proposition 5.3(iii) exhibits three pairwise separated prefixes for $n = 1$, where $a(1) = 2$.

Definition 4.2. A *certificate* for n consists of: a partial DFA B on M states with a labelling λ; a partial DFA A on m states; a partial map h from the states of B to those of A; a list P of m prefixes; and a list Z of suffixes. Its four *checks* are

(C1) the arrays have the declared sizes, $n \geq 1$, $M, m \geq 1$, $\lambda(q_0^B) = (0,0,0,0)$ and $h(q_0^B) = q_0^A$;
(C2) λ is admissible for n (Definition 3.3), checked at every state and letter;
(C3) h satisfies the hypotheses of Theorem 3.5, checked at every state and letter;
(C4) $|P| = m$, every prefix in P has a suffix in Z landing in $L_{n,0}$, and the bit vectors $([wz \in L_{n,0}])_{z \in Z}$, $w \in P$, are pairwise distinct.

Theorem 4.3. *If a certificate for n passes (C1)–(C3), then A recognises $L_{n,0}$; hence m lies in the size set of $L_{n,0}$, and $a(n) \leq m$.*

Proof. (C1)–(C2) and Theorem 3.4 give that B recognises $L_{n,0}$; (C3) and Theorem 3.5 transport this to A. (Formally `Cert.recognizes_small` and `Cert.mem_sizeSet_of_cert`.) □

Theorem 4.4. *If a certificate for n passes (C1)–(C4), then $a(n) = m$.*

Proof. Theorem 4.3 gives $m \in$ `SizeSet (Lang n)`; for any partial DFA on k states recognising $L_{n,0}$, (C4) and Theorem 4.1 give $m = |P| \leq k$. (Formally `Cert.sc_of_cert`.) □

Theorems 4.3 and 4.4 are proved once, for an arbitrary certificate; a value of $a(n)$ is then the evaluation of four Boolean functions on explicit data.

5. Exact values

Theorem 5.1. *For $1 \leq n \leq 16$ and for $n = 32$, $a(n)$ is as in Table 1. In particular $a(14) = 429$ and $a(32) = 2132$.*

Proof sketch. For each n a certificate in the sense of Definition 4.2 was generated as follows. The automaton B is the reachable part of the automaton on states (a', b', D, E) with transitions (3), a transition being deleted when the letter is blocked or the target (D, E) lies in Dead_n; its labelling is the identity. The automaton A is the quotient of B by iterated partition refinement, and h the class map, with the dead class sent to "undefined". The prefixes P are shortest words reaching each state of A, and Z is a greedy separating set of suffixes. The four checks were then evaluated by compiled code inside the proof assistant, and Theorem 4.4 applied. Every value of $a(n)$ asserted here rests on that exhaustive evaluation: (C2) and (C3) visit each of the $4M$ state–letter pairs, and (C4) evaluates $m \cdot |Z|$ memberships — at $n = 32$, $2132 \cdot 1696 \approx 3.6 \cdot 10^6$ of them. The certificates are not small: the one for $n = 32$ has $M = 4482$ labelled states. □

n	1	2	3	4	5	6	7	8	9
$a(n)$	2	10	23	40	59	85	114	146	181
M	8	28	55	90	134	188	249	317	394
$\lvert Z\rvert$	2	9	20	34	48	70	96	120	146

n	10	11	12	13	14	15	16	32
$a(n)$	224	269	318	371	429	489	552	2132
M	479	573	676	788	907	1035	1171	4482
$\lvert Z\rvert$	182	218	256	297	344	393	444	1696

TABLE 1. The certified values $a(n)$, the number M of states of the labelled automaton B of the certificate, and the number $|Z|$ of separating suffixes. The values $a(n)$ agree with [1, Remark 12] for $n \le 10$ and with [3] throughout.

Remark 5.2. The size of Z is about $0.8\,a(n)$ throughout Table 1, and this is not an artefact of the greedy choice: a state (D, E) accepts a suffix $s \times t$ of length k if and only if $F_k E + F_{k-1} D = n[s] - [t]$, so each suffix accepts the states on one line of the (D, E)-plane, and with every suffix of length at most 6 in the pool the best single suffix covers 19 of the 552 states at $n = 16$. The cost of (C4) is therefore $\Theta(a(n)^2)$, which is what limits the exact window; see Section 10. (Measured outside the formal development.)

Proposition 5.3 (controls)**.** (i) *$a(2) \neq 11$, $a(2) \neq 9$, and $a(3) \neq 10$: a too-large claim, a too-small claim and a wrong multiplier are refuted.*

(ii) *None of the four checks is vacuous. Taking the certificate for $n = 2$ and changing its multiplier to 3 fails (C2); redirecting one transition of A fails (C3); dropping one prefix, or the last suffix, fails (C4); mis-stating m fails (C1).*

(iii) *Liveness cannot be dropped from Theorem 4.1. For $n = 1$ the three prefixes ε, $(1,1)$, $(1,0)$ are pairwise separated by the suffixes ε, $(1,1)$, $(0,1)$, yet $a(1) = 2$: the prefix $(1,0)$ has no extension in $L_{1,0}$ at all, which is proved through the certified automaton for $n = 1$.*

Proof. (i) is immediate from $a(2) = 10$ and $a(3) = 23$ and the least-element property. (ii) and (iii) are Boolean evaluations on the stated data; for (iii), the membership table is 3×3, and the deadness of $(1,0)$ is read off the run of the certified 2-state automaton, which stops on that letter. □

6. The conjectured bound

Let $\varphi = (1 + \sqrt{5})/2$, so $\varphi^2 = (3 + \sqrt{5})/2$. Write $\mathrm{C}(n, a)$ for the real inequality $a \le 2n^2 + \varphi^2 n + 1$ — inequality (1) with a in place of $a(n)$, stated in the development over $\mathbb{R}$ with $\sqrt{5}$ exactly as the OEIS writes it (`ConjR`).

Proposition 6.1. *For all natural numbers n, a, with $T = 2a - 4n^2 - 3n - 2$,*

$$\mathrm{C}(n, a) \iff \big(T \le 0 \quad \vee \quad T^2 \le 5n^2\big).$$

Proof. Doubling, $\mathrm{C}(n, a)$ reads $2a \le 4n^2 + 3n + \sqrt{5}\,n + 2$, i.e. $T \le \sqrt{5}\,n$; for $T \le 0$ this holds, and for $T > 0$ it is equivalent to $T^2 \le 5n^2$. (Formally `conj_iff`; the right-hand side is `ConjZ n a`, a decidable statement about integers.) □

Two reading lemmas follow: if $a(n) = a$ and the integer form holds at (n, a) then $\mathrm{C}(n, a(n))$ (`conjR_of_sc`); and since the conjecture asserts an upper bound, a certified $m \in$ `SizeSet (Lang n)` with the integer form at (n, m) already gives $\mathrm{C}(n, a(n))$ whatever $a(n)$ is (`conjR_of_le`).

Theorem 6.2. *Inequality* (1) *holds for every $n \in \{1, \dots, 16, 32, 49, 50\}$. More precisely, for $n \le 16$ and $n = 32$ the certified value $a(n)$ of Theorem 5.1 satisfies $\mathrm{C}(n, a(n))$, and for $n = 49, 50$ every a with $a(n) = a$ satisfies $\mathrm{C}(n, a)$.*

Proof. For $n \le 16$ and $n = 32$, Proposition 6.1 at the pair $(n, a(n))$ is an integer computation, and `conjR_of_sc` applies. For $n = 49, 50$, the certified bounds $4926 \in$ `SizeSet (Lang 49)` and $5128 \in$

`SizeSet (Lang 50)` of Theorem 8.1 satisfy the integer form ($T = 99$, $T^2 = 9801 \le 12005 = 5 \cdot 49^2$; and $T = 104$, $T^2 = 10816 \le 12500$), and `conjR_of_le` applies. (Formally `conj1`, …, `conj16`, `conj32`, `conj49`, `conj50`.) □

Proposition 6.3 (tightness)**.** *At $n = 14$, the value 429 satisfies the integer form and 430 does not; at $n = 32$, 2132 satisfies it and 2133 does not. Accordingly* C(14, 430) *is false: one additional state at either n would refute* (1)*.*

Proof. At $(14, 430)$, $T = 2 \cdot 430 - 784 - 42 - 2 = 32 > 0$ and $T^2 = 1024 > 980 = 5 \cdot 14^2$; at $(32, 2133)$, $T = 2 \cdot 2133 - 4096 - 96 - 2 = 72 > 0$ and $T^2 = 5184 > 5120$. The falsity of C(14, 430) over $\mathbb{R}$ is then Proposition 6.1. □

Remark 6.4 (the range $n \le 601$; outside the formal development)*.* The pipeline that builds $M_{n,c}$, keeps its reachable part, adjoins a sink and minimises by partition refinement was run for every $1 \le n \le 601$ (1441 CPU-seconds in total, under 0.2 GB); its values reproduce the ten of [1, Remark 12] and the 48 of [3] exactly. Over that range:

(1) Inequality (1) is never violated.
(2) The slack $2n^2 + \varphi^2 n + 1 - a(n)$ attains its minimum 0.652476 at $n = 14$; the six smallest slacks are at $n = 14$ (0.652), 32 (0.777), 18 (1.125), 31 (1.159), 36 (1.249) and 15 (1.271), all within the first 48 terms. Exactly two n have slack below 1, namely 14 and 32 — the two values proved exactly in Theorem 5.1 and shown tight in Proposition 6.3. The published terms alone, read from $n = 1$ upward, suggest $n = 32$ as the point of danger; it is not the minimum, and there is none beyond.
(3) Beyond $n \approx 40$ the slack grows linearly: it is 60.4 at $n = 601$, about $0.101\,n$.
(4) Writing $s(n) = a(n) - 2n^2$, one has $s(n) \ge 0$ with equality only at $n = 1$; the first differences $s(n+1) - s(n)$ take the values $1, 2, 3, 4, 5$ with multiplicities $133, 146, 206, 104, 11$ and mean 2.5233; a least-squares line through $s(n)$ on $300 \le n \le 601$ has slope 2.5036 and intercept 5.70; the secant slopes of s over $[100, 200]$, $[200, 300]$, $[300, 400]$, $[400, 500]$, $[500, 601]$ are 2.560, 2.480, 2.490, 2.470, 2.564.

So the linear coefficient of $a(n)$, if it exists, is near 2.50 rather than $\varphi^2 = 2.618\ldots$, and the bound (1) is loose by about $0.12\,n$; the two near-misses at $n = 14$ and $n = 32$ are a small-n accident. The secant slopes show the constant is not pinned to better than about ± 0.05 by this range. None of this is a theorem about all n.

7. The reachable states of the source's automaton

The transition function and the initial state of $M_{n,c}$ (§2.4) do not mention c; only the accepting set does. Hence $\mathrm{Reach}(M_{n,c}) = \mathrm{Reach}(M_{n,0})$ for every c. In the development, a state $[a', b', d, d']$ is coded as $((2a' + b') \cdot 3n + (d + n)) \cdot 3n + (d' + n) < 36n^2$, the transition rule of §2.4 is written out on codes (`stepC`), and `reachCount n` is the size of the set found by a breadth-first search from the code of q_0 with an explicit visited array and a fuel bound of $36n^2 + 1$ steps.

Theorem 7.1. *For every $1 \le n \le 100$ and every c,*

$$|\,\mathrm{Reach}(M_{n,c})| \;=\; 6n^2 + 6n - 4.$$

In particular $|\,\mathrm{Reach}(M_{n,c})| < 36n^2$*, and* $6\,|\,\mathrm{Reach}(M_{n,c})| = 36n^2 + 36n - 24$*.*

Proof. An exhaustive computation: for each n from 1 to 100 the breadth-first search over the transition rule of [1, §3.2] is run to completion and its count compared with $6n^2 + 6n - 4$; the two consequences are the same searches compared with $36n^2$ and with $36n^2 + 36n - 24$. The independence of c is by inspection of the rule. (Formally `Reach.reach_eq`, `Reach.reach_lt_printed`, `Reach.reach_six`; the search is 100 evaluations of a self-contained program over machine integers.) □

Proposition 7.2 (control)**.** *The formula is exact and not merely a bound: $6n^2 + 6n - 3$ is not the count for every $1 \le n \le 100$. (Formally `Reach.reach_ne_off_by_one`.)*

Remark 7.3 (outside the formal development). The same breadth-first search, in a separate implementation, gives $6n^2+6n-4$ for every $1 \le n \le 601$. The count is not quadratic quadrant by quadrant: at $n = 6$ the four values of (a', b') contribute $77, 73, 51, 47$ states and at $n = 8$ they contribute $132, 126, 88, 82$, so a proof for all n must be about the total. We have no such proof; Conjecture 7.4 records the statement. Combined with the correctness of $M_{n,c}$ [1, Theorems 7 and 11] — a theorem of the source, not re-proved here — Theorem 7.1 says that for $n \le 100$ the construction of [1] yields a partial DFA of $6n^2 + 6n - 4$ states for $L_{n,0}$, so the explicit form of their $O(n^2)$ bound is $a(n) \le 6n^2+6n-4$ on that range, a factor of 6 below the $36n^2$ of their count. This is what the closing sentence of [1, Remark 12] asks for, although it remains far from the true $a(n) \approx 2n^2 + 2.5n$.

Conjecture 7.4. $|\operatorname{Reach}(M_{n,c})| = 6n^2 + 6n - 4$ for every $n \ge 1$ and every c.

8. Beyond the published terms

The OEIS entry ends at $a(48) = 4731$. The upper-bound half of a certificate, (C1)–(C3), costs only $O(M)$ evaluations, so it scales to larger n where the $\Theta(a(n)^2)$ separation check does not.

Theorem 8.1. $a(49) \le 4926$ *and* $a(50) \le 5128$*: partial DFAs on* 4926 *and* 5128 *states recognise* $L_{49,0}$ *and* $L_{50,0}$.

Proof. Certificates without a separation table, with $M = 10351$ and $M = 10771$ labelled states respectively, pass (C1)–(C3) by compiled evaluation; Theorem 4.3 applies. (Formally `up49` and `up50`.) □

Corollary 8.2. *Inequality* (1) *holds at* $n = 49$ *and* $n = 50$*, whatever the values* $a(49)$ *and* $a(50)$ *are. (Formally* `conj49`*,* `conj50`*; see Theorem 6.2.)*

Remark 8.3 (twelve new terms; outside the formal development). Two independent implementations — the $M_{n,c}$ pipeline of Remark 6.4, and the (a', b', D, E) construction of Section 3 followed by the same minimisation — agree on all 60 values $a(1), \dots, a(60)$, reproduce the 48 terms of [3], and give

$$a(49), \dots, a(60) = 4926,\ 5128,\ 5332,\ 5539,\ 5752,\ 5969,\ 6189,\ 6412,\ 6643,\ 6876,\ 7113,\ 7354.$$

The two constructions have different state counts before minimisation (656 against 479 at $n = 10$). The first continues to $a(600) = 721512$ and $a(601) = 723916$. The certified statements are the two upper bounds of Theorem 8.1; the matching lower bounds at $n = 49, 50$, and the exact values for $17 \le n \le 31$, $33 \le n \le 48$ and $n \ge 49$, are computed, not proved, for the reason given after Table 1: the separation check at $n = 49$ would evaluate about $0.8 \cdot 4926^2 \approx 1.9 \cdot 10^7$ memberships, measured at about 20 minutes in a single compiled evaluation, and was not run.

9. Verification

Every theorem, proposition and corollary of Sections 3–8 is formally verified in Lean 4 [12] with Mathlib [13], in a development of thirteen modules with no `sorry`; the remarks labelled "outside the formal development" are not part of it. The reasoning layer — the update (3), Lemma 3.2, Theorems 3.4, 3.5, 4.1, 4.3, 4.4, Proposition 6.1 and the two reading lemmas — is proved by ordinary induction and arithmetic and depends on no axioms beyond `propext`, `Classical.choice` and `Quot.sound` (Theorem 3.5 on `propext` and `Quot.sound` only). What is delegated to compiled evaluation, through Lean's `native_decide`, is exactly the finite searches: for each of the seventeen values of Theorem 5.1 the four checks (C1)–(C4), one evaluation each; for the two bounds of Theorem 8.1 the three checks (C1)–(C3); the hundred breadth-first searches behind each statement of Theorem 7.1 and Proposition 7.2; and the perturbed certificates of Proposition 5.3(ii). The reachability module imports nothing from Mathlib, so Theorem 7.1 and its control depend on their single compiled evaluation and on nothing else. Proposition 6.3, Proposition 5.3(iii)'s separation table and its non-liveness are decided by the kernel's own evaluator (`decide`), with no compiled code; the deadness of the third prefix there goes through the certified automaton for $n = 1$ and its compiled checks. The certificate data are stored as decimal strings and parsed at evaluation time; the wall-clock cost of the compiled checks on a loaded machine was about two minutes for all of $n \le 16$, nine minutes for $n = 32$, seven seconds for $n = 49, 50$ and two

minutes for Theorem 7.1. Appendix A lists the formal statements verbatim. Repository reference to be added at submission.

10. Open questions

(1) *Prove Conjecture 7.4.* A bijective description of Reach($M_{n,c}$) would turn Theorem 7.1 into a statement for all n and answer the closing sentence of [1, Remark 12] in closed form; the quadrant counts of Remark 7.3 say the description must treat the four values of (a', b') together.

(2) *The linear term.* Is $a(n) - 2n^2 - \kappa n$ bounded for some constant κ, and if so what is κ? Remark 6.4 puts it near 2.50, below φ^2; the OEIS bound (1) would then hold for all n with room to spare, but neither the existence of κ nor (1) itself is proved. The one-tape value $2n^2$ of [4] is exactly the quadratic term; whether their argument can be made to see the two-tape linear term is open.

(3) *A cheaper lower-bound certificate.* The separation check (C4) costs $\Theta(a(n)^2)$. A levelled certificate — the partitions $P_0 \supseteq \cdots \supseteq P_R$ of Moore's refinement, each level checked to be a function of the signature of the previous one — would cost $O(R \cdot a(n))$ and push the exact window well past $n = 100$; the obstruction is certifying the "function of" clause without a quadratic table.

(4) $c \neq 0$. Everything here is stated for $L_{n,0}$; $L_{n,c}$ needs only the accepting set changed, and Reach($M_{n,c}$) is already known to be independent of c.

Appendix A. The formal statements

The following are the statements of the development, verbatim, that the results above are pinned to; `Lang`, `SizeSet` and `SC` are the language, the size set and the least-element predicate of §2, `PDFA m` a partial DFA on m states, `PWord` a word over $\Sigma_2 \times \Sigma_2$, `Cert` the certificate of Definition 4.2 with its checks `okStruct`, `okLab`, `okHom`, `okMN` for (C1)–(C4), and `certN` the certificate for $n = N$.

Language, size set, state complexity, and the conjecture.

```
def Lang (n : ℕ) (w : PWord) : Prop :=
  pvalid w = true ∧ fval (sndW w) = n * fval (fstW w)
def SizeSet (L : PWord → Prop) : Set ℕ := {m | ∃ A : PDFA m, Recognizes A L}
def SC (L : PWord → Prop) (m : ℕ) : Prop := IsLeast (SizeSet L) m
def ConjR (n a : ℕ) : Prop :=
  (a : ℝ) ≤ 2 * (n : ℝ) ^ 2 + ((1 + Real.sqrt 5) / 2) ^ 2 * (n : ℝ) + 1
def ConjZ (n a : ℕ) : Prop :=
  let T : ℤ := 2 * (a : ℤ) - 4 * (n : ℤ) ^ 2 - 3 * (n : ℤ) - 2
  T ≤ 0 ∨ T ^ 2 ≤ 5 * (n : ℤ) ^ 2
def Dead (n : ℕ) (d e : ℤ) : Prop := (2 * n ≤ d ∧ 3 * n ≤ e) ∨ (d ≤ -2 ∧ e ≤ -3)
```

Theorem 4.1 (Myhill–Nerode for partial acceptors).

```
theorem length_le_of_sep {m : ℕ} (A : PDFA m) (L : PWord → Prop) (hA : Recognizes A L)
    (ws : List PWord)
    (hlive : ∀ w ∈ ws, ∃ z, L (w ++ z))
    (hsep : ws.Pairwise (fun x y => ∃ z, ¬ (L (x ++ z) ↔ L (y ++ z)))) :
    ws.length ≤ m
theorem length_le_of_cert {n m : ℕ} (A : PDFA m) (hA : Recognizes A (Lang n))
    (ws Z : List PWord)
    (hlive : ws.all (fun w => Z.any (fun z => memB n (w ++ z))) = true)
    (hsep : nodupN (ws.map (fun w => maskL Z (fun z => memB n (w ++ z)))) = true) :
    ws.length ≤ m
```

Theorems 4.3 and 4.4 (certificate soundness).

```
theorem recognizes_small (C : Cert) (hs : C.okStruct = true) (hl : C.okLab = true)
    (hh : C.okHom = true) : Recognizes (C.small (C.hm hs)) (Lang C.n)
theorem mem_sizeSet_of_cert (C : Cert) (hs : C.okStruct = true) (hl : C.okLab = true)
    (hh : C.okHom = true) : C.m ∈ SizeSet (Lang C.n)
theorem sc_of_cert (C : Cert) (hs : C.okStruct = true) (hl : C.okLab = true)
    (hh : C.okHom = true) (hn : C.okMN = true) : SC (Lang C.n) C.m
```

Theorem 5.1 (exact values).

```
theorem a1 : SC (Lang 1) 2          theorem a9  : SC (Lang 9) 181
theorem a2 : SC (Lang 2) 10         theorem a10 : SC (Lang 10) 224
theorem a3 : SC (Lang 3) 23         theorem a11 : SC (Lang 11) 269
theorem a4 : SC (Lang 4) 40         theorem a12 : SC (Lang 12) 318
theorem a5 : SC (Lang 5) 59         theorem a13 : SC (Lang 13) 371
theorem a6 : SC (Lang 6) 85         theorem a14 : SC (Lang 14) 429
theorem a7 : SC (Lang 7) 114        theorem a15 : SC (Lang 15) 489
theorem a8 : SC (Lang 8) 146        theorem a16 : SC (Lang 16) 552
theorem a32 : SC (Lang 32) 2132
```

Proposition 6.1, the reading lemmas, and Theorem 6.2.

```
theorem conj_iff (n a : ℕ) : ConjR n a ↔ ConjZ n a
theorem conjR_of_sc {n a : ℕ} (h : SC (Lang n) a) (hz : ConjZ n a) : ConjR n a
theorem conjR_of_le {n a m : ℕ} (h : SC (Lang n) a) (hm : m ∈ SizeSet (Lang n))
    (hz : ConjZ n m) : ConjR n a
theorem conj1 : ConjR 1 2           theorem conj9  : ConjR 9 181
theorem conj2 : ConjR 2 10          theorem conj10 : ConjR 10 224
theorem conj3 : ConjR 3 23          theorem conj11 : ConjR 11 269
theorem conj4 : ConjR 4 40          theorem conj12 : ConjR 12 318
theorem conj5 : ConjR 5 59          theorem conj13 : ConjR 13 371
theorem conj6 : ConjR 6 85          theorem conj14 : ConjR 14 429
theorem conj7 : ConjR 7 114         theorem conj15 : ConjR 15 489
theorem conj8 : ConjR 8 146         theorem conj16 : ConjR 16 552
theorem conj32 : ConjR 32 2132
theorem conj49 : ∀ a, SC (Lang 49) a → ConjR 49 a
theorem conj50 : ∀ a, SC (Lang 50) a → ConjR 50 a
```

Theorem 8.1 (certified upper bounds).

```
theorem up49 : (4926 : ℕ) ∈ SizeSet (Lang 49)
theorem up50 : (5128 : ℕ) ∈ SizeSet (Lang 50)
```

Theorem 7.1 and Proposition 7.2 (`reachCount n` is the breadth-first count of §7; `printedCount n = 2 * 2 * (3 * n) * (3 * n)`).

```
theorem reach_eq :
    (List.range' 1 100).all (fun n => reachCount n == 6 * n ^ 2 + 6 * n - 4) = true
theorem reach_lt_printed :
    (List.range' 1 100).all (fun n => decide (reachCount n < printedCount n)) = true
theorem reach_six :
    (List.range' 1 100).all
      (fun n => 6 * reachCount n + 24 == printedCount n + 36 * n) = true
theorem reach_ne_off_by_one :
    (List.range' 1 100).all (fun n => reachCount n == 6 * n ^ 2 + 6 * n - 3) = false
```

Propositions 5.3 and 6.3 (controls); `ws3 = [[], [(true, true)], [(true, false)]]` and `zs3 = [[], [(true, true)], [(false, true)]]`.

```
theorem not_sc_2_11 : ¬ SC (Lang 2) 11
theorem not_sc_2_9 : ¬ SC (Lang 2) 9
theorem not_sc_3_10 : ¬ SC (Lang 3) 10
theorem bad_n : ({cert2 with n := 3} : Cert).okLab = false
theorem bad_hom : ({cert2 with sstep := cert2.sstep.set! 0 9} : Cert).okHom = false
theorem bad_pre : ({cert2 with pre := cert2.pre.drop 1} : Cert).okMN = false
theorem bad_suf : ({cert2 with suf := cert2.suf.dropLast} : Cert).okMN = false
theorem bad_m : ({cert2 with m := 9} : Cert).okStruct = false
theorem ws3_sep :
    nodupN (ws3.map (fun w => maskL zs3 (fun z => memB 1 (w ++ z)))) = true
theorem ws3_not_live :
    ws3.all (fun w => zs3.any (fun z => memB 1 (w ++ z))) = false
theorem ws3_third_dead : ∀ z : PWord, ¬ Lang 1 ([(true, false)] ++ z)
theorem conj_14_ok : ConjZ 14 429
theorem conj_14_tight : ¬ ConjZ 14 430
theorem conj_32_tight : ConjZ 32 2132 ∧ ¬ ConjZ 32 2133
theorem conjR_false_430 : ¬ ConjR 14 430
```

References

[1] D. Moradi, N. Rampersad and J. Shallit, *Complexity of Linear Subsequences of Fibonacci-Automatic Sequences*, arXiv:2603.21645v1, 23 March 2026, cs.FL (cross-listed cs.DM, math.CO). Numbered statements cited here (Lemmas 2 and 4, Theorems 7 and 11, Remark 12, §§2.2 and 3.2) follow the PDF compiled from the v1 e-print, the only version as of 3 September 2026.

[2] D. Moradi, *State Complexity of Linear Relations and Linear Subsequences of Automatic Sequences*, Master's thesis, University of Waterloo, 2026 (UWSpace, 88 pp.; Remark 46 on p. 67 of the PDF, printed page 57), `hdl.handle.net/10012/23044`.

[3] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, sequence A372846 (J. Shallit, 4 July 2024; 48 terms; entry revision of 28 April 2026), `oeis.org/A372846`, consulted 3 September 2026.

[4] É. Charlier, N. Rampersad, M. Rigo and L. Waxweiler, *The minimal automaton recognizing $m\mathbb{N}$ in a linear numeration system*, INTEGERS 11B (2011), #A4.

[5] D. Moradi, N. Rampersad and J. Shallit, *Complexity of Linear Subsequences of k-Automatic Sequences*, arXiv: 2512.10017, first posted 10 December 2025, cs.FL.

[6] Korea Superintelligence Labs, *Exact msd-first state complexity of shifts of the Fibonacci word*, companion manuscript, 2026.

[7] J.-P. Allouche and J. Shallit, *Automatic Sequences: Theory, Applications, Generalizations*, Cambridge University Press, Cambridge, 2003.

[8] J. Shallit, *The Logical Approach to Automatic Sequences: Exploring Combinatorics on Words with* **`Walnut`**, London Mathematical Society Lecture Note Series 482, Cambridge University Press, 2022, doi:10.1017/9781108775267.

[9] H. Mousavi, *Automatic Theorem Proving in* **`Walnut`**, arXiv:1603.06017.

[10] E. Zeckendorf, *Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas*, Bull. Soc. Roy. Sci. Liège 41 (1972), 179–182.

[11] C. G. Lekkerkerker, *Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci*, Simon Stevin 29 (1952), 190–195.

[12] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, in: Automated Deduction — CADE 28, Lecture Notes in Computer Science, Springer, 2021, pp. 625–635, doi:10.1007/978-3-030-79876-5_37.

[13] The mathlib Community, *The Lean mathematical library*, in: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020), ACM, 2020, pp. 367–381.