A 211-state Turing machine for ZFC consistency

Shivam Patel

12 November 2026

Abstract

We construct a two-symbol Turing machine with 211 working states and prove in first-order Peano arithmetic (PA) that it halts on a blank tape if and only if ZFC has a finite proof of contradiction. Starting from Reshetnikov's 215-state machine, we eliminate three control states by changing its proof-search commands and save a fourth by replacing a read-only tape scanner with a writable one. A Metamath Zero certificate establishes the equivalence for the literal transition table. A separate Lean formalization checks the same machine. Consequently, if ZFC is consistent, it proves no numerical upper bound on the step-count Busy Beaver function at 211.

1 Introduction

An exact Busy Beaver value decides the blank-tape halting problem for every machine of a given size. An explicit machine that searches for a contradiction in set theory therefore gives a quantitative form of Gödel's second incompleteness theorem [1]. Yedidia and Aaronson constructed a 7,910-state machine to obtain such a bound by way of a combinatorial independence result [7]. O'Rear subsequently constructed a 748-state machine that searches directly for a contradiction in ZF [1]. Our starting point is Reshetnikov's 215-state machine K_{215} [6]. Glazer gave a proof in Peano arithmetic that K_{215} halts precisely when ZFC is inconsistent [3].

We use a single tape indexed by $\mathbb{Z}$, with alphabet $\{0, 1\}$. An n-state machine has working states $0, \ldots, n-1$ and a separate halt state n. A transition writes one bit, moves one cell left or right, and selects the next state. The initial tape is zero everywhere, with state and head position both zero. Write $\mathrm{Halt}(M)$ for the arithmetical assertion that some finite run from this initial configuration reaches the halt state. Let $\mathrm{Prf}_{\mathsf{ZFC}}(p, \ulcorner \bot \urcorner)$ mean that p codes a finite first-order derivation of falsity from Zermelo–Fraenkel set theory with the axiom of choice (ZFC).

Theorem 1. *For the machine K_{211} of Appendix A,*

$$\mathsf{PA} \vdash \big(\mathrm{Halt}(K_{211}) \longleftrightarrow \exists p\, \mathrm{Prf}_{\mathsf{ZFC}}(p, \ulcorner \bot \urcorner)\big).$$

Following Radó [5], let $\mathrm{BB}(n)$ be the maximum number of steps taken by a halting n-state machine on the blank tape, including the transition into the halt state. We use the running-time version throughout.

Corollary 2. *If ZFC is consistent, then for every standard $b \in \mathbb{N}$, $\mathsf{ZFC} \nvdash \mathrm{BB}(211) \leq \overline{b}$. If N is the least n for which ZFC cannot prove the true value of $\mathrm{BB}(n)$, then $6 \leq N \leq 211$.*

Proof. Assume ZFC is consistent and fix b. By Theorem 1, K_{211} does not halt. Its first b steps form a particular finite computation, so ZFC can prove that K_{211} has not halted by step b. A proof in ZFC of the proposed upper bound would then give a proof in ZFC that K_{211} never halts. Since ZFC interprets the arithmetic of Theorem 1, it would prove Con(ZFC), contrary to the second incompleteness theorem [4]. The lower endpoint follows from the known values through BB(5) = 47,176,870 [2]. □

The three changes to the instruction language affect which commands a candidate code denotes. To justify the reduction we construct a new compiler from finite derivations to candidate codes. A separate simulation argument accounts for the change to the tape scanner.

2 Four state reductions

The binary table of K_{215} implements a register program which enumerates words of commands in a numerical proof calculus. Formula codes 0 and 1 represent truth and falsity, respectively. The calculus uses the pairing and implication codes

$$P(a,b) = \frac{(a+b)(a+b+1)}{2} + b, \qquad I(a,b) = 4P(a,b).$$

At a command boundary the proof store is (t, w, d, h), where t is the top formula, w the stack of saved formulas, d a numerical counter and h the cached formula. An empty stack returns zero on a pop. We retain the numerical calculus and its calibration to ZFC [6, 3].

Two jump states. The old DROP pushes t and copies h to the top. The command NIP halts if $h = 1$, and otherwise pops a formula into the cache. Their composition is thus a guarded exchange S. If $h \neq 1$ it acts by

$$S(t, w, d, h) = (h, w, d, t).$$

Two jumps in K_{215} skip this exchange. Bypassing the first redirects the read-one transition of native state 37 from 117 to 134. Bypassing the second redirects the read-one transition of native state 163 from 162 to 174. Each bypass removes the sole incoming edge of the deleted state. The resulting machine K_{213} interprets tag 0 as NIP, tag 1 as S, tag 2 as modus ponens followed by S, and higher tags as axiom generation followed by S. Where a compiled word needs the original modus-ponens or axiom effect, one further exchange restores the formula registers, unless it already detects falsity.

One axiom-check state. The axiom command AX$(c; x, y)$ forms $A = d + x$ and $B = h + y$. Its axiom bundle is permitted when c differs from both A and B. Otherwise the gated output is truth. Move the test $c = A$ to the end of the branch and redirect the read-one transition of native state 146 from 96 to 77. This removes state 96. On $c \neq A$ the command retains the result of the first test, including zero when $c = B$. On $c = A$ it falls through to modus ponens and reads a further field z. Here both the counter and cache are zero. If u denotes the prospective top, the modus-ponens comparison succeeds exactly when

$$I(u, z) = 0 \quad \Longleftrightarrow \quad u = z = 0.$$

Failure clears the top. Success also gives top zero and pushes one additional truth code. Thus the prospective top cannot escape as an inadmissible axiom. This edit gives a 212-state machine, denoted by C_{212}.

One scanner state. The last reduction changes the binary interpreter without changing the register program of C_{212}. An eight-state read-only scanner is replaced by the seven-state routine in Table 1. It can write temporary markers. At entry the head is immediately left of a known one. If k zeros separate this marker from the next one to its left, $0 \le k \le 7$, the scanner restores the tape and restarts immediately left of the second marker. For $k = 0, \dots, 7$ the respective costs are $1, 4, 11, 14, 23, 26, 39, 42$ steps. If the eight cells at offsets $-7, \dots, 0$ from the entry head are zero, it restores them and reaches the existing return routine in 58 steps. An interior gap of exactly eight zeros can distinguish the two scanners on an arbitrary tape. Correctness of the substitution therefore uses the actual program addresses and initialization, as proved in Section 4.

Local state	0	1	2	3	4	5	6
Read 0	`0L1`	`0R3`	`0L0`	`1R3`	`0R5`	`0L0`	`0RU`
Read 1	`1L0`	`1R2`	`0L6`	`1L4`	`1L1`	`0L2`	`0L5`

Table 1: The writable scanner. An entry gives the bit written, the head move, and the next local state. `U` is the external return routine.

3 Correctness of the proof search

Let $D(a)$ mean that formula code a is derivable in the numerical calculus. We use the calibration $D(1) \leftrightarrow \exists p \, \mathrm{Prf}_{\mathsf{ZFC}}(p, \ulcorner \bot \urcorner)$ established for the parent construction. The two logical obligations for the new register program are that an arbitrary detection is sound and that every derivation of 1 can be detected by a word the program actually enumerates.

Lemma 3. *From the register seed* $(1, 0, 0, 0, 2, 0, 0, 0, 0)$*, the program of* K_{211} *halts if and only if* $D(1)$.

Proof. At this seed the next candidate is 1, the counter is 2, and the other registers are zero. Maintain the invariant that the top and cache are derivable and all formulas saved on the stack are derivable. Stack operations preserve this invariant. A successful modus-ponens command with field x tests $I(t, d + x) = h$ and obtains $d + x$ from its two stored premises. The axiom command supplies the permitted instances, and the additional fallthrough on $c = A$ can introduce only truth, as shown above. Hence a detection of $h = 1$ implies $D(1)$.

For the converse, first use the virtual commands S, NIP, MP, and AX, where the last two have their restored effects. A derivation of a yields a finite word with effect

$$(t, w, 0, 0) \longmapsto (a, w, 0, 0) \tag{1}$$

for every t, w, allowing earlier detection. For example, a block that saves the top formula is

$$\mathsf{AX}(0; 0, 0)\,, \quad \mathsf{NIP}^{9}\,, \quad \mathsf{MP}(0).$$

Here the exponent denotes repetition and the commands are virtual. The axiom command produces top zero and pushes eight axiom instances, followed by the code 2 of the quantification of truth and the old top t. After nine pops the cache holds 2 and t is at the head of the stack. Since $I(0, 0) = 0 \neq 2$, the final comparison fails, leaving $(0, t :: w, 0, 0)$, where $t :: w$ denotes the stack with t prepended.

The construction of (1) is by induction on the derivation. An axiom block selects the requested entry from the generated list. A generalization block saves the premise and selects its quantified formula. For modus ponens, first construct $I(u, v)$ and save it on the stack. The block for u leaves this stack intact by (1). Popping the implication into the cache places the operands in the order required by $\mathsf{MP}(v)$. A final normalization restores the stack, counter and cache. This explains why the uniformity in w is needed even for a single fixed derivation.

It remains to encode these blocks in the altered language. When $c = A$, supply $z = 0$ for the extra field. If $c \neq B$, the prospective bundle is a negation, with code $I(v, 1) = 4P(v, 1) > 0$ for some v. The comparison fails and leaves the axiom command's stack intact. If $c = B$, both inputs to the comparison are zero, and one extra NIP removes the truth code that was pushed. When $c \neq A$, an explicit exchange suffices to restore the virtual axiom command.

The choice of fields must still work in the store left by earlier candidate words. For a free parameter Y, begin with the four raw commands

$$\mathsf{AX}(0; 1, Y)\,, \quad S\,, \quad \mathsf{MP}(1)\,, \quad \mathsf{AX}(0; 0, 0).$$

Since $d + 1 > 0$ for every incoming d, the first command never takes the extra field. This prefix leaves top, counter and cache zero, independently of the inherited stack. Repeated padding commands then place a known prefix of theorem codes above the inherited stack. A virtual command consumes at most one pre-existing head, and a corrective NIP consumes only the head introduced by its own equality branch. For a proof word of length m, more than m known heads suffice. The compiler can therefore fix all branch choices without inspecting the unknown tail.

Apply this construction to a derivation of 1 and append an exchange and a cache test. The resulting word detects falsity from every incoming store. Its initial candidate code is at least Y, so successful candidate codes are cofinal in $\mathbb{N}$. Every unsuccessful candidate finishes because each positive continuation is decremented before its fields are unpaired, and unpairing cannot increase the remaining code. The serial enumeration therefore reaches a detecting word unless a detection has occurred earlier. □

4 Realization by the binary machine

A dispatcher frame represents an instruction address and nine natural-valued registers. It contains an address followed by zero padding, a header, the register blocks and a nonempty finite list of spare blocks. A register block for value v consists of a zero followed by $v + 1$ ones. Write $C \sim (i, r)$ if the binary configuration C represents instruction i with register vector r in such a frame. The relation allows the head position, additional padding and spare blocks to vary.

Lemma 4. *Starting on the blank tape, K_{211} halts if and only if the register program halts from instruction zero with register vector* $(1, 0, 0, 0, 2, 0, 0, 0, 0)$.

Proof. The literal initialization takes 613,447 steps. It reaches dense state 22, the dispatcher, with head at -5, extra gap 123 and 168 spare blocks of value one. The complete tape at this point represents instruction zero and the stated register vector.

For every nonhalting register step $(i, r) \to (j, r')$ and every $C \sim (i, r)$, the table admits a positive finite segment $C \to^{+} C'$ with $C' \sim (j, r')$. The halt instruction instead reaches the binary halt label. These assertions are verified for all 4,746 instruction occurrences. There are 1,900 increments, 1,424 conditional decrements, 1,326 continues, 95 breaks and one halt. Splitting each decrement on zero versus positive input gives 6,170 cases.

Each case is a symbolic tape calculation. A known bit is handled by its literal transition. A self-loop crossing a homogeneous run is justified by induction on its length. For a traversal of spare registers, one first proves the action on a single arbitrary block with its exterior held fixed, and then inducts on the finite list of blocks. Thus the finite case split imposes no bound on register values or tape length.

Concatenation now gives a halting binary execution for every halting register execution. Conversely, an infinite register execution yields nonhalting dispatcher frames at unbounded binary times, since each segment has positive length. With the halt label made absorbing, the binary run cannot halt between two such frames. □

Proof of Theorem 1. Lemma 4 identifies blank-tape halting with register-program halting from the seed of Lemma 3. The latter lemma and the calibration of $D(1)$ identify this with the existence of a finite ZFC refutation. Both equivalences are theorems of PA. Their composition proves the displayed statement for the literal table. □

Formal verification. The Metamath Zero certificate proves `k211Literal_halts_iff_ZFC` over the same 26 foundational axiom declarations as Glazer's PA formalization. The separate Lean 4.32.1 theorem uses the same 422 transitions and a syntactic ZFC derivability predicate. Its kernel proof depends on propositional extensionality, classical choice and quotient soundness. The accompanying files identify the exact specifications and provide the sources and replay procedures for both proofs.

Acknowledgments. The interpreter and numerical calculus build on work by Mira, Jeff Ketchersid, Andrew J. Wade, Elliot Glazer, Vladimir Reshetnikov and the ProveIt contributors.

References

[1] S. Aaronson. The Busy Beaver frontier. *ACM SIGACT News* 51(3) (2020), 32–54.

[2] The bbchallenge Collaboration. Determination of the fifth Busy Beaver value. arXiv:2509.12337, 2025, revised 2026.

[3] E. Glazer. *K215 halts iff ZFC is inconsistent.* Metamath Zero proof in Peano arithmetic, 2026. Archived PA certificate, Zenodo record 23872596.

[4] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik* 38 (1931), 173–198.

[5] T. Radó. On non-computable functions. *Bell System Technical Journal* 41 (1962), 877–884. doi:10.1002/j.1538-7305.1962.tb00480.x.

[6] V. Reshetnikov and the ProveIt contributors. *K215: 215 working states for ZFC consistency.* K215 transition table and construction, 2026. Formal library used for the Lean calibration.

[7] A. Yedidia and S. Aaronson. A relatively small Turing machine whose behavior is independent of set theory. *Complex Systems* 25(4) (2016), 297–327.

A The transition function of K_{211}

An entry wDq' means "write w, move one cell in direction D, and enter state q'." The columns headed 0 and 1 correspond to the symbol read. The initial state is 0 and the halt state is 211. The numbering here is the dense numbering of the certified machine.

q	0	1	q	0	1	q	0	1	q	0	1
0	1R1	1R0	53	0R43	1R71	106	0R132	1R133	159	0R127	1R167
1	0R2	0R0	54	0R69	1R72	107	0R56	0R75	160	0R61	1R104
2	1R3	1L4	55	0L73	1L54	108	0R109	1R55	161	0R81	1R126
3	1L5	1L6	56	0L74	0R0	109	0R76	0R93	162	0R168	1R169
4	0L5	1L4	57	0R75	1R57	110	0R43	0L11	163	0R52	1L39
5	0L7	1L4	58	0R76	0R59	111	0R55	1R90	164	0R124	1R124
6	0L8	0L9	59	0R77	1R59	112	1R0	1R92	165	0R84	1R156
7	1R10	0L7	60	0R45	1R78	113	0R81	1R73	166	0R170	1R156
8	0L7	0L8	61	0R79	1R68	114	0L134	1R105	167	0R114	1R171
9	1L6	1L9	62	0R80	1R211	115	0R135	1R87	168	0R99	1R172
10	0R10	0R11	63	0L81	1R53	116	0R136	1R137	169	0R173	1R174
11	0L12	0R11	64	0L82	1L83	117	0R107	1R119	170	0R104	1R61
12	1L13	0L13	65	0R84	1R85	118	0L138	0L13	171	0R102	1R175
13	0L13	1L14	66	0R59	1R66	119	0R110	0L8	172	0R165	1R176
14	0L15	1L14	67	0R56	0R77	120	0R139	1R105	173	0R177	1R178
15	0R16	1R17	68	0R86	1R53	121	0R140	1R137	174	0R179	1R180
16	1R16	1L18	69	0R76	0R66	122	0R136	1R141	175	0R181	1R104
17	0L14	0L19	70	0L27	1R87	123	0R142	1R128	176	0R124	1R166
18	0R20	1L15	71	0R88	0L12	124	0R109	1R109	177	0R182	1R183
19	0R21	0L20	72	0R89	0L90	125	0R143	1R144	178	0L184	1R185
20	0L14	0L17	73	0R91	1R90	126	0R105	1R104	179	0L186	1L83
21	0R21	1R22	74	1R74	1L13	127	0R99	1R124	180	0R178	1R145
22	0R23	1R24	75	0R50	1R75	128	0R127	1R99	181	0R82	1R178
23	0R25	1R26	76	0L74	0R92	129	0R64	1R109	182	0R187	1R159
24	0R27	1R28	77	0R93	1R77	130	0R145	1R33	183	0R186	1R188
25	0R29	0L5	78	0R36	1R94	131	0R51	1R139	184	0R179	1R189
26	0R30	1R31	79	0R73	1R95	132	0R63	1R146	185	0R82	1R99
27	0R32	1R33	80	0R96	1R97	133	0R147	1R148	186	0R168	1R82
28	0R34	1R35	81	0R98	1R99	134	0R149	1R150	187	0R124	1R179
29	0L36	0R37	82	0L100	1L101	135	0L36	0R49	188	0R128	1R190
30	0R38	1R39	83	0R102	1R103	136	0R151	1R152	189	0R179	1R191
31	0R38	1R40	84	0R104	1R105	137	0R55	1R140	190	0R192	1R184
32	0R41	1R42	85	0R106	1R80	138	0R86	1R151	191	0R193	1R194
33	0R43	1R44	86	0R107	1R108	139	0R43	1R153	192	0R195	1R196
34	0R45	1R46	87	0R109	0L12	140	0R135	1L70	193	0R185	1R178
35	0R40	1R47	88	0L41	1L86	141	0R55	1R154	194	0R197	1R193
36	0L28	1R48	89	0R110	1R111	142	0R155	1R156	195	0R198	1R181
37	0R49	1R37	90	0R40	0L12	143	0R157	1R158	196	0R142	1R199
38	0L36	0R50	91	0R56	0R93	144	0L159	1R160	197	0R179	1R200
39	0R29	1L51	92	0R112	1R92	145	0R45	1L138	198	0R101	1R201
40	0L52	1L53	93	0R37	1R93	146	0R99	1R161	199	0R202	1R203
41	0R29	1R54	94	0R113	1R114	147	0R144	1R134	200	0R204	1R204
42	0R55	1R39	95	0R115	1R116	148	0R111	1R51	201	0R179	1R205
43	0R56	0R57	96	0R117	1R43	149	0R156	1R162	202	0R100	1R206
44	0R58	0L12	97	0R118	1R119	150	0R80	1R118	203	0R207	1R194
45	0R56	0R59	98	0R32	1R120	151	0R91	1R88	204	0R160	1L208
46	0R60	1R61	99	0R121	1R122	152	0L52	1R163	205	0R64	1R209
47	0R62	1R63	100	0R104	1R123	153	0R152	0L12	206	0R170	1R204
48	0R64	1R65	101	0R124	1R125	154	0R164	0L12	207	0R126	1R210
49	0R12	1R49	102	0R126	1R127	155	0R105	1R61	208	0R144	1R165
50	0R66	1R50	103	0R128	1R129	156	0R99	1L103	209	0R102	1R64
51	0R67	1L68	104	0R79	1R130	157	0R61	1R83	210	0R126	1R158
52	0R69	1R70	105	0R79	1R131	158	0R165	1R166			