hexagon:2611.00474v1

# AN UPPER BOUND FOR THE CONNECTIVE CONSTANT OF THE SQUARE LATTICE

TEJASVI SINGH TOMAR

Abstract. We prove that the connective constant $\mu$ of the square lattice $\mathbb{Z}^2$, the exponential growth rate of the number of $n$-step self-avoiding walks, satisfies $\mu \le 2.6586513$. The previous best upper bound, $\mu \le 2.662342426$, is due to Couronné. Like the bounds of Pönitz and Tittmann and of Couronné, ours comes from a finite automaton that remembers a bounded part of the past of the walk. The new idea is to count bridges. Since they can all be preceded by the same infinite history, the automaton may treat as visited the unvisited cells that no infinite self-avoiding continuation can reach, and three tests on the current state and the next step recognize such cells. The bound is checked in exact integer arithmetic and proved in the Lean proof assistant.

## 1. Introduction

Let $c_n$ be the number of $n$-step self-avoiding walks on the square lattice $\mathbb{Z}^2$ that start at the origin. Since $c_{n+m} \le c_n c_m$, the limit

$$\mu = \lim_{n\to\infty} c_n^{1/n}$$

exists [5]. It is called the connective constant of $\mathbb{Z}^2$. Its value is not known; numerical estimates give $\mu \approx 2.63815853032790$ [7].

Upper bounds for $\mu$ come from counting walks with a bounded memory. Alm [1] proved $\mu \le 2.69576$, and Noonan [13] $\mu \le 2.6939$. Pönitz and Tittmann [14] built finite automata whose states are the last steps of the walk, with the steps that can no longer matter removed, reaching $\mu \le 2.679192495$. Couronné [3] added several simplification rules to the automaton of Pönitz and Tittmann and, considering loops of length up to 26, obtained $\mu \le 2.662342426$. This was the best upper bound so far. Lower bounds come from counting irreducible bridges; in this way Jensen [8] obtained $\mu > 2.625622$. On the hexagonal lattice the connective constant is known to be exactly $\sqrt{2+\sqrt{2}}$ [4].

**Theorem 1.1.** *The connective constant of the square lattice satisfies*

$$\mu \le \frac{105432502045487}{39656386815908} = 2.658651241600\ldots < 2.6586513.$$

Our automaton reads bridges instead of walks. This loses nothing, because for the number $b_n$ of $n$-step bridges that start at the origin the Hammersley–Welsh theorem [6] gives $\mu = \lim b_n^{1/n}$. Here a self-avoiding walk is a bridge if its first cell lies strictly to the left of all its other cells and its last cell lies in its rightmost column. A bridge that starts at the origin, preceded by the cells $(x, 0)$ with $x < 0$ and followed by translates of itself, each starting where the previous one ends, forms a two-sided infinite self-avoiding path. We give every bridge this half-line as its history. The automaton may then treat as visited every cell that the walk has not visited but that no

2020 *Mathematics Subject Classification.* 82B41; 05A15, 60K35, 68Q45, 68V05, 68V20.

*Key words and phrases.* self-avoiding walk, connective constant, bridges, finite automata, computer-assisted proof, formal verification.

two-sided infinite self-avoiding path extending the history and the walk so far can visit, since the walk will never visit it. We call such cells dead.

Figure 1 shows how dead cells arise. The state there is a path of 13 cells ending at the origin, which is the current position; the next step goes to the cell $t$. Each cell marked a has three of its four neighbours in the state, so a path that entered it could not leave it. The cell marked b is enclosed by the state and $t$. The cell marked c is connected to infinity only through the cell $w$; without $w$ it is cut off both from infinity and from the neighbours of $t$ outside the state. These are the three tests of Definition 3.1. Section 4 shows that the cells these tests find are dead, although they need not find every dead cell. In general a state is a set of cells, written as a lattice path ending at the current position, each of which is visited or dead. Our construction also uses simplification rules of [3] that put unvisited cells into a state; in our certificate every such cell passes one of the three tests, as the checkers verify.

The automaton is given by a finite certificate: its states, a child or the symbol $\varnothing$ (no child) for every state and step, and nonnegative integer weights. Distinct $n$-step bridges give distinct paths of length $n$ from the start state (Theorem 4.1). The number of paths of length $n$ grows at most like $\lambda^n$, where $\lambda$, the ratio of the certificate, is the largest ratio of the total weight of the children of a state to its weight, over the states of positive weight (Definition 3.2 and Lemma 5.1). Definition 3.2 also restricts the states of weight zero. Hence $\mu \le \lambda$. For our certificate, $\lambda$ is the fraction of Theorem 1.1.

Section 2 introduces the history and the bridges, and Section 3 the automata with dead cells and their certificates. Section 6 explains how we built our certificate, of order 30, and compares its ratio with the values of [3]. Section 7 presents two independently written checkers that verify it in exact integer arithmetic, and Section 8 two formal proofs of Theorem 1.1 in Lean 4, which start from the definition of $c_n$ and formalize every step of the proof.

## 2. Bridges

We write $e_1 = (1, 0)$ and $e_2 = (0, 1)$, and $x(z)$ for the first coordinate of $z \in \mathbb{Z}^2$. An $n$-step self-avoiding walk is a sequence $\omega_0, \dots, \omega_n$ of distinct points of $\mathbb{Z}^2$ with $|\omega_i - \omega_{i-1}| = 1$. It is a bridge if

$$x(\omega_0) < x(\omega_i) \le x(\omega_n) \qquad (1 \le i \le n).$$

Let $b_n$ be the number of $n$-step bridges with $\omega_0 = 0$. The sequence $(b_n)$ is supermultiplicative. Hammersley and Welsh [6] proved that $c_n \le b_{n+1} e^{O(\sqrt{n})}$. Hence (see also [9, Section 3.1])

$$\mu = \lim_{n\to\infty} b_n^{1/n}. \tag{2.1}$$

Let $H = \{(x, 0) : x \le 0\}$. If $\omega$ is a bridge with $\omega_0 = 0$, then $\omega_i \notin H$ for $i \ge 1$, because $x(\omega_i) \ge 1$. Concatenating $\omega$ with translates of itself gives an infinite self-avoiding path that starts with $\omega$ and stays in the half-plane $\{x \ge 1\}$ after its first step. Hence the two-sided sequence that runs through $H$ from $x = -\infty$ to 0 and then follows this path is a two-sided infinite self-avoiding path. We call $H$ the history of the bridge. We call the points of $\mathbb{Z}^2$ cells. For $0 \le k \le n$, a *continuation* of $\omega$ at time $k$ is a sequence $(\gamma_i)_{i\in\mathbb{Z}}$ of distinct cells with $|\gamma_i - \gamma_{i-1}| = 1$ for all $i$, $\gamma_i = (i, 0)$ for $i \le 0$ and $\gamma_i = \omega_i$ for $0 \le i \le k$. It *visits* a cell $z$ after time $k$ if $\gamma_i = z$ for some $i > k$. By the above, continuations exist for every $k$. A continuation at time $j$ is also a continuation at every time $k \le j$.

## 3. Automata with dead cells

3.1. **States and steps.** Let $D_4$ be the group of the eight symmetries of $\mathbb{Z}^2$ that fix the origin. A *state* is a self-avoiding lattice path $S = (s_0, \dots, s_{m-1})$ with $m \ge 2$, $s_{m-1} = 0$ and $s_{m-2} = -e_1$. We write $|S|$ for its set of cells. The origin is the current position, and the last step of the path

is $+e_1$. The three *steps* of a state are $t \in \{e_1, e_2, -e_2\}$; the fourth neighbour $-e_1$ of the origin is a cell of the state. The *start state* $S_{\mathrm{st}}$ is the straight path $s_i = (i - K/2, 0)$, $0 \le i \le K/2$, of $K/2 + 1$ cells, for an even integer $K \ge 2$, which we call the order of the automaton. The order enters the definitions below only through $S_{\mathrm{st}}$; it mainly governs the construction of the automaton (Section 6).

For a finite set $E \subset \mathbb{Z}^2$, the complement $\mathbb{Z}^2 \setminus E$ has exactly one infinite connected component (for 4-connectivity). A cell of $\mathbb{Z}^2 \setminus E$ is *connected to infinity in* $\mathbb{Z}^2 \setminus E$ if it lies in this component. This is the case if and only if it is connected, within $\mathbb{Z}^2 \setminus E$, to a cell outside the bounding box of $E$.

**Definition 3.1.** Let $S$ be a state, $t$ a step with $t \notin |S|$, and $E = |S| \cup \{t\}$. A cell $z \notin E$ is *dead for* $(S, t)$ if

(a) at least three of the four neighbours of $z$ lie in $|S|$; or
(b) $z$ is not connected to infinity in $\mathbb{Z}^2 \setminus E$; or
(c) there is a cell $w \notin E \cup \{z\}$ such that, in $\mathbb{Z}^2 \setminus (E \cup \{w\})$, the cell $z$ is connected neither to infinity nor to any neighbour of $t$ outside $E \cup \{w\}$.

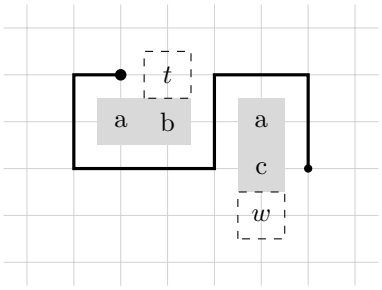



FIGURE 1. A state $S$ of 13 cells, drawn as a path from its first cell (small dot) to the origin (large dot), entered with the last step $+e_1$, and the step $t = e_1$. Each marked cell is dead by the test of Definition 3.1 with the same letter, test (c) with the cell $w$.

### 3.2. **Certificates.**

**Definition 3.2.** A *certificate* of order $K$ consists of a finite set $\mathcal{S}$ of states that contains $S_{\mathrm{st}}$, an integer $V_S \ge 0$ for each $S \in \mathcal{S}$, and, for each $S \in \mathcal{S}$ and each step $t$, an entry that is either the symbol $\varnothing$ or a state $S' \in \mathcal{S}$, such that the following hold.

(C1) If the entry for $(S, t)$ is $\varnothing$, then $t \in |S|$, or $t$ is not connected to infinity in $\mathbb{Z}^2 \setminus |S|$.
(C2) If the entry for $(S, t)$ is a state $S'$ and $t \notin |S|$, then there is $g \in D_4$ such that every cell of the set $g(|S'|) + t$ lies in $E = |S| \cup \{t\}$ or is dead for $(S, t)$.
(C3) $V_{S_{\mathrm{st}}} > 0$; every state with $V_S = 0$ has only children with $V = 0$; and there is no sequence $S_1, \dots, S_r = S_1$ $(r \ge 2)$ of states with $V = 0$ in which each $S_{i+1}$ is a child of $S_i$.

Here the *children* of $S$ are the states $S'$ that occur in its entries. The *ratio* of the certificate is

$$\lambda = \max\Big\{ V_S^{-1} \sum_t V_{S'_t} : S \in \mathcal{S},\ V_S > 0 \Big\},$$

where the sum runs over the steps $t$ whose entry is a state $S'_t$.

An entry $S'$ for a step $t \in |S|$ is allowed and need not satisfy any condition. Such an entry is never used by a walk and can only increase the counts below. The set $g(|S'|) + t$ in (C2) contains both $t = g(0) + t$ and the cell $g(-e_1) + t$, which is the image of the second-to-last cell

of $S'$; so (C2) requires the cell in the excluded direction $-e_1$ of the next state to lie in $E$ or to be dead. The *automaton graph* of a certificate is the directed graph on $\mathcal{S}$ with one edge from $S$ to $S'$ for each step of $S$ whose entry is $S'$. A *path* of length $n$ in it is a sequence of $n$ edges in which each edge starts at the state where the previous one ends; states and edges may repeat.

## 4. Soundness

**Theorem 4.1.** *Let a certificate of order $K$ be given. For every $n \geq 1$, $b_n$ is at most the number of paths of length $n$ in the automaton graph that start at $S_{\mathrm{st}}$.*

*Proof.* Fix an $n$-step bridge $\omega$ with $\omega_0 = 0$. For $0 \leq k \leq n$ the *past* at time $k$ is $P_k = H \cup \{\omega_0, \dots, \omega_k\}$. A cell is *dead at time $k$* if no continuation of $\omega$ at time $k$ visits it after time $k$. The cells of $P_k$ are dead at time $k$, because the cells of a continuation are distinct. A cell that is dead at time $k$ is dead at every later time $j$, because a continuation at time $j$ is a continuation at time $k$, and a visit after time $j$ is a visit after time $k$.

We define states $S^k \in \mathcal{S}$ and isometries $\Phi_k(z) = h_k(z) + \omega_k$ of $\mathbb{Z}^2$, with $h_k \in D_4$, such that

$$\text{every cell of } \Phi_k(|S^k|) \text{ is dead at time } k. \tag{4.1}$$

Let $S^0 = S_{\mathrm{st}}$ and $\Phi_0 = \mathrm{id}$. Then $\Phi_0(|S^0|) \subset H \subset P_0$, so (4.1) holds for $k = 0$.

Suppose that $S^k$ and $\Phi_k$ satisfy (4.1), and let $k < n$. Put $t = h_k^{-1}(\omega_{k+1} - \omega_k)$, a neighbour of the origin. Let $\gamma$ be a continuation of $\omega$ at time $n$. It is also a continuation at time $k$, and it visits $\omega_{k+1} = \gamma_{k+1}$ after time $k$, so $\omega_{k+1}$ is not dead at time $k$. By (4.1), $t \notin |S^k|$. In particular $t \neq -e_1$, so $t$ is a step of $S^k$. The cells $\gamma_i$ with $i \geq k+1$ are not dead at time $k$, by the definition of dead cells, so they avoid $\Phi_k(|S^k|)$. Hence the cells $\Phi_k^{-1}(\gamma_i)$, $i \geq k+1$, form an infinite self-avoiding path that starts at $t$ and avoids $|S^k|$, and $t$ is connected to infinity in $\mathbb{Z}^2 \setminus |S^k|$. By (C1), the entry of $(S^k, t)$ is a state $S'$. Let $g$ be the first element, in a fixed list of the elements of $D_4$, that satisfies (C2) for this entry. This choice makes the construction deterministic. Let $S^{k+1} = S'$ and $\Phi_{k+1}(z) = \Phi_k(g(z) + t)$. Then $h_{k+1} = h_k g$ and $\Phi_{k+1}(0) = \Phi_k(t) = \omega_{k+1}$.

We show (4.1) for $k+1$. Let $E = |S^k| \cup \{t\}$. Every cell of $\Phi_k(E)$ is dead at time $k+1$: the cells of $\Phi_k(|S^k|)$ were dead at time $k$, and $\Phi_k(t) = \omega_{k+1} \in P_{k+1}$. Let $z$ be a cell of $g(|S'|) + t$. If $z \in E$, then $\Phi_k(z)$ is dead at time $k+1$. Otherwise $z$ is dead for $(S^k, t)$ by (C2), and we show that $\Phi_k(z)$ is dead at time $k+1$. Let $\delta$ be a continuation of $\omega$ at time $k+1$, and suppose that $\delta$ visits $\Phi_k(z)$ after time $k+1$. The cells $\delta_i$ with $i > k+1$ are not dead at time $k+1$, so they avoid $\Phi_k(E)$. Apply $\Phi_k^{-1}$, which is an isometry of $\mathbb{Z}^2$ that maps $\omega_{k+1}$ to $t$. The images of the cells $\delta_i$, $i > k+1$, form a self-avoiding path $\delta'$ that starts at a neighbour of $t$, avoids $E$, passes through $z$ and then goes to infinity. In case (a), at most one neighbour of $z$ lies outside $|S^k|$. If there is none, then $\delta'$ can neither start at $z$, which is then not a neighbour of $t$, nor enter $z$. Otherwise let $u$ be this neighbour. If $z$ is the first cell of $\delta'$, then $z$ is a neighbour of $t$, so $u = t$, all four neighbours of $z$ lie in $E$, and $\delta'$ cannot leave $z$. If not, $\delta'$ enters $z$ from a neighbour outside $E$, which can only be $u$ (so $u \neq t$), and it cannot leave $z$, because the other neighbours of $z$ lie in $E$ and $\delta'$ has already visited $u$. In case (b), $\delta'$ goes from $z$ to infinity in $\mathbb{Z}^2 \setminus E$, which is impossible. In case (c), let $w$ be the cell of the test and $Q$ the component of $z$ in $\mathbb{Z}^2 \setminus (E \cup \{w\})$. The first cell of $\delta'$ is a neighbour of $t$ outside $E$, so it is not in $Q$ (it may be $w$), $Q$ is finite, and every cell adjacent to $Q$ but not in $Q$ lies in $E \cup \{w\}$. Hence $\delta'$ passes through $w$ before it reaches $z$ and again after it leaves $z$, which contradicts self-avoidance. In each case we have a contradiction, so $\Phi_k(z)$ is dead at time $k+1$. This proves (4.1) for $k+1$.

Let $t_k$ be the step $t$ used at time $k$. The edges of the automaton graph given by $S^k$ and $t_k$, for $0 \leq k < n$, form a path of length $n$ that starts at $S_{\mathrm{st}}$. The bridge can be recovered from this path: $\omega_{k+1} = \Phi_k(t_k)$, and $\Phi_{k+1}$ is determined by $\Phi_k$ and by the symmetry $g$, which depends

only on $S^k$, $t_k$ and the entry of $(S^k, t_k)$. Hence distinct bridges give distinct paths, which proves the theorem. □

*Remark* 4.2. Two monotonicity facts are used in the proof. First, every cell of $\Phi_k(E)$ is dead at time $k+1$, so a continuation at time $k+1$ avoids $\Phi_k(E)$ after time $k+1$. Second, if $z$ is dead for $(S^k, t)$, then $\Phi_k(z)$ is dead at time $k+1$, although the tests (a)–(c) look only at $E$: the case analysis uses only that $\delta'$ is an infinite self-avoiding path that starts at a neighbour of $t$ and avoids $E$. The cells of the past outside $\Phi_k(E)$ play no role; self-avoidance is used at cells outside $E$ (the neighbour $u$ in case (a), the cell $w$ in case (c)).

## 5. Counting

**Lemma 5.1.** *Let a certificate have ratio $\lambda$, and let $L$ be the number of its states with $V = 0$. The number of paths of length $n$ in the automaton graph that start at $S_{\mathrm{st}}$ is at most*

$$V_{S_{\mathrm{st}}}\,\lambda^n + 3^{L+1}\,V_{S_{\mathrm{st}}} \sum_{k=0}^{n-1} \lambda^k.$$

*If no state has $V = 0$, it is at most $V_{S_{\mathrm{st}}}\lambda^n$.*

*Proof.* For $k \geq 0$ and a state $S$ with $V_S > 0$, let $W_k(S)$ be the number of paths of length $k$ from $S_{\mathrm{st}}$ to $S$ that visit only states with $V > 0$. Then $\sum_S W_k(S)V_S \leq \lambda^k V_{S_{\mathrm{st}}}$. This holds for $k = 0$, and

$$\sum_S W_{k+1}(S)V_S \leq \sum_{S'} W_k(S') \sum_t V_{S'_t} \leq \lambda \sum_{S'} W_k(S')V_{S'},$$

where the outer sums run over states with $V > 0$ and the inner sum runs over the steps $t$ of $S'$ whose entry is a state $S'_t$, as in the definition of $\lambda$. Since $V_S \geq 1$ when $V_S > 0$, the number of such paths of length $k$ is at most $\lambda^k V_{S_{\mathrm{st}}}$. A path that leaves the states with $V > 0$ does so at its first edge into a state with $V = 0$, and from then on it only visits states with $V = 0$, by (C3). These states form an acyclic graph with out-degrees at most 3, so the rest of the path has length less than $L$, and there are at most $3^L$ possible rests for each first edge. The first edge into a state with $V = 0$ follows a path of some length $k \leq n-1$ that stays in $V > 0$, and there are at most 3 choices for it. This gives the bound. □

*Proof of Theorem 1.1.* By Theorem 4.1 and Lemma 5.1, $b_n \leq C\,n \max(\lambda, 1)^n$ for the ratio $\lambda$ of any certificate, with $C$ independent of $n$. By (2.1), $\mu \leq \max(\lambda, 1)$. By the computation of Section 7, the files described there define a certificate of order 30 in the sense of Definition 3.2, and its ratio $\lambda$ is the fraction in Theorem 1.1, which is larger than 1. □

## 6. Construction of the automaton

This section describes how we found the certificate.

Our construction program starts from the start state and explores the states reachable by steps. For a state $S$ and a step $t$ it lists several candidate children: subsets of $E = |S| \cup \{t\}$ together with dead cells, written as paths ending at $t$ and placed by a symmetry so that they end at the origin with last step $+e_1$, as in Section 3. The size of a path $(s_0, \ldots, s_{m-1})$ is $m - 1 + |s_{m-1} - s_0|_1$, the length of the loop obtained by closing it with a shortest lattice path. The candidates come from rules of the kind introduced by Pönitz and Tittmann [14] and Couronné [3]:

- old cells are forgotten when the size exceeds the order $K$ (Couronné's parameter $k$ plays the role of $K$); at order 30 the size of the child that keeps all cells may exceed $K$ by a bonus of at most 8. The bonus is read from one of two tables at the index $(m - 1 - |s_{m-1} - s_0|_1)/2$,

which measures how far the path is from a shortest path between its ends, and a test on the shape of the child chooses the table; the README of the archive gives the tables with the build command;
- cells cut off by bridge configurations in the sense of [3], which are not related to the bridges of Section 2, are recognized as dead;
- loops sealed by the walk are treated as blocked regions;
- steps into trapped regions get no child.

Couronné also uses a rule based on the enclosure of the oldest cell of a state (Section 4.2 of the arXiv version of [3]). Our section numbers and values from [3] refer to this version. Our program does not use this rule, and gives a step no child only in the two cases allowed by (C1). The small-loop rule of Couronné (Section 3.4 of [3]) requires that neither end of a loop sealed except for one gap cell is the current position, whereas our program allows such an end to be the current position, provided that no neighbour of the current position lies inside the loop; we call this condition the guard. A variant of our program without the guard produces children that violate (C2), as the rejected certificates of Appendix A show.

The program explores the states twice. In the second pass, when it shortens a candidate child, it may stop at any state found in the first pass. It then alternates power iteration, which gives approximate positive weights, with a new choice, for every state and step, of the candidate child of smallest weight. The weights in the certificate are those of the last power iteration, computed before the last new choice of children; that choice can only lower the sums. The ratio $\lambda$ is computed exactly from these weights.

Table 1 compares the ratio of our certificate with the values reported in [3]. It does not isolate the effect of dead cells, which would need certificates built with and without them at the same order.

TABLE 1. Upper bounds from automata: the values reported in [3] and the ratio of our certificate, truncated to nine decimals and accepted by both checkers of Section 7.

| | order | states | bound |
|---|---|---|---|
| [3] | 24 | 79,510,267 | 2.664196283 |
| [3] | 26 | 430,365,791 | 2.662342426 |
| Theorem 1.1 | 30 | 24,887,440,008 | 2.658651241 |

## 7. The certificate and its verification

The checks below concern the certificate itself, so the construction of Section 6 need not be trusted. The certificate for Theorem 1.1 has order $K = 30$ and 24,887,440,008 states, none with $V = 0$. It is stored in three binary files: the states as 64-bit keys that encode their paths, the entries as indices into the list of states, and the weights as double-precision numbers $v_S$, from which the checker computes $V_S = \lfloor 2^{60} v_S \rfloor$ exactly. The README of the archive describes the format.

The checker reads the three files and verifies the following, for every state $S$ and every step $t$.
- The keys are strictly increasing and decode to self-avoiding paths that end at the origin with last step $+e_1$, the start state is present, every $v_S$ lies in $[0, 1]$, and every index is smaller than the number of states.
- If the entry is $\varnothing$, then $t \in |S|$, or a flood fill from $t$ in $\mathbb{Z}^2 \setminus |S|$ does not reach the boundary of the bounding box of $|S|$ enlarged by one cell. This is (C1).

- If the entry is a child $S'$, the checker tries the eight symmetries $g \in D_4$ (the symmetry is not stored) and accepts the entry if for one of them every cell of $g(|S'|) + t$ lies in $E$ or passes one of the tests (a)–(c) of Definition 3.1. It does this also when $t \in |S|$, where Definition 3.2 requires nothing. For (b) it uses a flood fill in $\mathbb{Z}^2 \setminus E$ to the boundary of the bounding box of $E$ enlarged by one cell. For (c) it tries as $w$ the cells of one shortest path in $\mathbb{Z}^2 \setminus E$ from $z$ to that boundary; since (b) has failed, such a path exists. A cell outside the bounding box of $E$ stays connected to infinity when any one other cell is removed, so a flood fill with $w$ removed decides whether $z$ is connected to infinity in $\mathbb{Z}^2 \setminus (E \cup \{w\})$. Every cell $w$ for which (c) holds lies on every path from $z$ to the boundary, in particular on the chosen one. This is (C2).
- It computes the ratio $\lambda$ as an exact fraction in integer arithmetic and checks (C3).

The checker accepts the certificate, and its ratio is the fraction of Theorem 1.1.

A second, independently written checker also accepts the certificate, with the same ratio; Appendix A describes it and the tests of both checkers.

## 8. Formal verification

Using the Lean 4 proof assistant [11] and the Mathlib library [10], we also proved Theorem 1.1, with the certificate of Section 7. The formal theorem has no hypotheses:

```
theorem connective_constant_le_unconditional :
    ∃ μ : ℝ, Tendsto (fun n : ℕ => (sawCount n : ℝ) ^ ((1 : ℝ) / n)) atTop (𝓝 μ) ∧
      μ ≤ (105432502045487 : ℝ) / 39656386815908 ∧ μ < 2.6586513
```

It states that $c_n^{1/n}$ converges to a limit $\mu$, that $\mu \le 105432502045487/39656386815908$ and that $\mu < 2.6586513$. Here `sawCount n` is $c_n$: a walk is given by its sequence of $n$ steps in $\{e_1, -e_1, e_2, -e_2\}$, its $n+1$ points are the partial sums of its steps, the first being the origin, and `sawCount n` is the number of step sequences whose points are distinct.

```
def walk {n : ℕ} (s : Fin n → Fin 4) (i : Fin (n + 1)) : Cell :=
  ∑ j : Fin n, if (j : ℕ) < (i : ℕ) then dir (s j) else 0

def sawCount (n : ℕ) : ℕ :=
  (univ.filter fun s : Fin n → Fin 4 => Function.Injective (walk s)).card
```

Here `Cell` is $\mathbb{Z} \times \mathbb{Z}$ and `dir` maps $0, 1, 2, 3$ to $e_1, -e_1, e_2, -e_2$. Apart from `Cell`, `dir`, `walk` and `sawCount`, the statement uses only definitions from Lean's core library and from Mathlib.

The formal proof follows the paper. Its definition of a certificate is Definition 3.2 with (C3) replaced by the condition $V_S > 0$ for all $S$; for such certificates it contains Theorem 4.1 and Lemma 5.1. It also contains the submultiplicativity $c_{m+n} \le c_m c_n$ and the existence of $\mu$ by Fekete's lemma. In the proof of Theorem 4.1 it uses one continuation $\gamma$ of the bridge, the one through the history, the bridge and its translates (Section 2). It replaces (4.1) by the statement that $\gamma$ does not visit $\Phi_k(|S^k|)$ after time $k$, proved by the case analysis above, and (2.1) by a weaker form of the Hammersley–Welsh inequality,

$$c_n \le (n+1)^{2\lfloor\sqrt{2n}\rfloor+1}\, b_{n+1}, \tag{8.1}$$

which suffices because the factor is $e^{o(n)}$. The proof splits the walk at the first time at which its first coordinate is smallest and unfolds each of the two parts by reflections. The injection records the split time and the times of the at most $\lfloor\sqrt{2n}\rfloor$ reflections in each part. With a step $e_1$ put in front, this gives an injective map from walks to pairs of such a record and a bridge of $n+1$ steps, and there are at most $(n+1)^{2\lfloor\sqrt{2n}\rfloor+1}$ records. With the bound for $b_n$, this inequality yields $\mu \le \lambda$.

The certificate is checked by a program written in Lean, and a Lean proof shows that every certificate it accepts satisfies the formal definition with ratio at most $\lambda$. In this first formal proof, the check runs as compiled code, split into theorems proved by `native_decide`, so its

trusted base includes the Lean compiler and interpreter, the C compiler that Lean uses and the Lean runtime. Besides the standard axioms `propext`, `Classical.choice` and `Quot.sound`, the theorem depends on one auxiliary axiom for each compiled check, and the rest of the proof, including the soundness of the checking program and inequality (8.1), is checked by Lean's kernel.

A second formal proof of Theorem 1.1, with the same statement, is in the directory `lean_k30e/` of the archive. Since it depends only on `propext`, `Classical.choice` and `Quot.sound`, it needs no code generated by Lean's compiler, and its trusted base is Lean's kernel, including the kernel's built-in arithmetic on natural numbers. A proof of this kind for the weaker bound $\mu < 2.6614721$, from an earlier certificate of order 26, is archived at doi:10.5281/zenodo.23636439. Appendix A describes both checks.

## 9. Concluding remarks

The bound of Theorem 1.1 remains about 0.02 above the numerical estimate of [7]. In [3], raising the order from 24 to 26 lowers the bound by about 0.002 and multiplies the number of states by about 5.4, and our automaton of order 30 already has about $2.5 \cdot 10^{10}$ states, for which the first checker of Section 7 needs about 1.3 terabytes of memory. The method is thus limited by the size of the automaton. Lowering the bound further at a manageable size therefore needs rules or tests that lower the ratio at a given order, for instance tests that recognize more dead cells.

## Appendix A. Details of the checks

A.1. **The second checker and the tests.** The checking logic of the second checker of Section 7 was first written from the description of the checked conditions, before its author read the first checker; the version used differs from this first version only in its output and in three changes that can only cause a rejection, which the README of its directory lists. It uses flood fills on rows stored as bit masks, and for (c) it tries as $w$ only the cells of the component of $z$ in $\mathbb{Z}^2 \setminus E$ that share a side or a corner with a cell of $E$. No valid $w$ is missed. Test (b) has shown that $z$ is connected to infinity, and removing a cell $w$ leaves $z$ connected to infinity in two cases: if none of the eight cells around $w$ lies in $E$, because these cells then connect the four neighbours of $w$; and if $w$ lies outside the component of $z$, because removing $w$ then does not change that component. The two checkers report the same numbers of entries $\varnothing$, of entries with a child and of dead cells of each kind, although they list $D_4$ in different orders, on which the counts of dead cells depend.

A program in Python also checks certificates of order 10, 12 and 14. They were built by the same construction program with the same rules and the default values of its other parameters, without the tables of bonuses used at order 30. Unlike the two checkers, it tries as $w$ every cell of the bounding box of $E$ enlarged by one cell, and it gives the same ratios as the two checkers. Both checkers accept the certificates of order 10 to 16 built in this way.

Both checkers reject certificates that must fail (those of orders 12, 14 and 16 built without the guard of Section 6, and the accepted certificate of order 16 with a child removed or children redirected), and both report a larger ratio after a weight of that certificate is halved.

A.2. **The formal proofs.** The checking program of the first formal proof checks the conditions of Definition 3.2 for every state, with positive weights and the ratio of Theorem 1.1. Every claim that a cell is not connected to infinity, or in test (c) to a neighbour of $t$, is checked again on the region that the flood fill returns: this region must contain the cell and lie in a bounded box, every unblocked neighbour of its cells must lie in it, and in test (c) it must contain no unblocked neighbour of $t$; so errors in the flood fills or in the search for $w$ cannot lead to an acceptance.

Neither the construction of the automaton (Section 6) nor the largest ratio is computed in Lean; the program checks only that $V_S^{-1} \sum_t V_{S'_t} \le \lambda$ for every state $S$.

The certificate enters as data, converted into records that Lean compiles into string constants; the checking program checks whatever records it is given, and the theorem follows if the check succeeds.

The general parts of the proof (the soundness of the checking program, the cover of the states by the compiled checks, Theorem 4.1, Lemma 5.1, the bound $\mu \le \lambda$ from the bound for $b_n$, and inequality (8.1)) depend only on the standard axioms.

The second formal proof checks the certificate inside Lean's kernel. Instead of trying the eight symmetries for a child and the three tests for a dead cell, its checking program reads hints: the symmetry to use, and the test (a), (b) or (c) that applies, with the cell $w$ for test (c). The hints are computed beforehand by compiled code but are not trusted, since a wrong hint can only make the check fail; the README of the directory `lean_k30e/` describes the structure of this proof and its build. For both formal proofs, the general parts were also accepted by two further implementations of the Lean kernel, nanoda [12] and lean4lean [2].

## Data availability

The two checkers, the program in Python, the construction program, the programs for the tests of Appendix A, and the Lean files of both formal proofs of Section 8 with the programs that generate their data files are archived at doi:10.5281/zenodo.23715897, together with the list of the units of the second formal proof, the SHA-256 checksums of their compiled files, and the compiled public files of its data modules, part modules and root. The three files of the certificate were compressed with xz, the file of entries in four pieces and the file of weights in five; the ten compressed files are archived in five records, doi:10.5281/zenodo.23574056, doi:10.5281/zenodo.23574058, doi:10.5281/zenodo.23574060, doi:10.5281/zenodo.23574062 and doi:10.5281/zenodo.23574064.

Three earlier certificates of ours, of orders 26 and 28, built by an earlier version of the construction program, are also archived, and both checkers accept them. The one of order 28 with 5,792,114,594 states is at doi:10.5281/zenodo.23551536 (keys and weights) and at doi:10.5281/zenodo.23551538 (entries). The other two are at doi:10.5281/zenodo.23548806; they have fewer than $2^{32} - 1$ states and store the entries as 32-bit indices, with $2^{32} - 1$ for $\varnothing$. The second checker uses 32-bit indices exactly when there are fewer than $2^{32} - 1$ states. The first checker takes the width from the size of the file of entries. The repository `https://github.com/tejasvi/connective-constant-bound/tree/v8` (commit `9a451bfefdf4`) also holds the programs and the Lean files.

## Use of AI tools

Generative AI tools were used to find and prove the results, to write the programs and the Lean formalization, and to draft the text. The author reviewed the arguments, the programs and the formal statements, and takes responsibility for the content.

## References

[1] S. E. Alm, *Upper bounds for the connective constant of self-avoiding walks*, Combin. Probab. Comput. **2** (1993), 115–136.

[2] M. Carneiro, *Lean4Lean: verifying a typechecker for Lean, in Lean*, arXiv:2403.14064, 2024.

[3] O. Couronné, *New upper bound for the connective constant for square-lattice self-avoiding walks*, J. Stat. Phys. **192** (2025), 153; arXiv:2211.16146.

[4] H. Duminil-Copin and S. Smirnov, *The connective constant of the honeycomb lattice equals* $\sqrt{2+\sqrt{2}}$, Ann. of Math. (2) **175** (2012), 1653–1665.

[5] J. M. Hammersley and K. W. Morton, *Poor man's Monte Carlo*, J. Roy. Statist. Soc. Ser. B **16** (1954), 23–38.
[6] J. M. Hammersley and D. J. A. Welsh, *Further results on the rate of convergence to the connective constant of the hypercubical lattice*, Quart. J. Math. Oxford Ser. (2) **13** (1962), 108–110.
[7] J. L. Jacobsen, C. R. Scullard and A. J. Guttmann, *On the growth constant for square-lattice self-avoiding walks*, J. Phys. A **49** (2016), 494004.
[8] I. Jensen, *Improved lower bounds on the connective constants for two-dimensional self-avoiding walks*, J. Phys. A **37** (2004), 11521–11529.
[9] N. Madras and G. Slade, *The Self-Avoiding Walk*, Birkhäuser, Boston, 1993.
[10] The mathlib Community, *The Lean mathematical library*, in: Proc. 9th ACM SIGPLAN Int. Conf. Certified Programs and Proofs (CPP 2020), 2020, 367–381.
[11] L. de Moura and S. Ullrich, *The Lean 4 theorem prover and programming language*, in: Automated Deduction – CADE 28, Lecture Notes in Comput. Sci. **12699**, Springer, 2021, 625–635.
[12] `nanoda_lib`, a type checker for Lean 4, `https://github.com/ammkrn/nanoda_lib`.
[13] J. Noonan, *New upper bounds for the connective constants of self-avoiding walks*, J. Stat. Phys. **91** (1998), 871–888.
[14] A. Pönitz and P. Tittmann, *Improved upper bounds for self-avoiding walks in* $\mathbb{Z}^d$, Electron. J. Combin. **7** (2000), R21.

*ORCID*: `https://orcid.org/0000-0003-4668-3639`