

Lagrange's Theorem for Binary Squares

Parthasarathy Madhusudan
Department of Computer Science
Thomas M. Siebel Center for Computer Science
201 North Goodwin Avenue
Urbana, IL 61801-2302
USA
`madhu@cs.uiuc.edu`

Dirk Nowotka
Department of Computer Science
Kiel University
D-24098 Kiel
Germany
`dn@informatik.uni-kiel.de`

Aayush Rajasekaran and Jeffrey Shallit
School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1
Canada
`{arajasekaran, shallit}@uwaterloo.ca`

October 13, 2017

Abstract

We prove, using a decision procedure based on finite automata, that every natural number > 686 is the sum of at most 4 natural numbers whose canonical base-2 representation is a *binary square*, that is, a string of the form xx for some block of bits x .

1 Introduction

Additive number theory is the study of the additive properties of integers [8]. In particular, an *additive basis of order h* is a subset $S \subseteq \mathbb{N}$ such that every natural number is the sum of h members, not necessarily distinct, of S . The principal problem of additive number theory is to determine whether a given subset S is an additive basis of order h for some h , and if so, to determine the smallest value of h . There has been much research in the area, and deep techniques, such as the Hardy-Littlewood circle method, have been developed to solve problems.

One of the earliest results in additive number theory is Lagrange's famous theorem that every natural number is the sum of four squares [3, 7]. In the terminology of the previous paragraph, this means that $S = \{0^2, 1^2, 2^2, 3^2, \dots\}$ forms an additive basis of order 4. The celebrated problem of Waring (1770) (see, e.g., [2, 12, 13]) is to determine the corresponding least order $g(k)$ for k 'th powers. Since it is easy to see that numbers of the form $4^a(8k + 7)$ cannot be expressed as the sum of three squares, it follows that $g(2) = 4$. It is known that $g(3) = 9$ and $g(4) = 19$.

In a variation on this concept we say that $S \subseteq \mathbb{N}$ is an *asymptotic additive basis of order h* if every *sufficiently large* natural number is the sum of h members, not necessarily distinct, of S . The function $G(k)$ is defined to be the least asymptotic order for k 'th powers. From above we have $G(2) = 4$. It is known that $G(14) = 16$, and $4 \leq G(3) \leq 7$. Despite much work, the exact value of $G(3)$ is currently unknown.

In this paper we consider a variation on Lagrange's theorem. Instead of the ordinary notion of the square of an integer, we consider "squares" in the sense of formal language theory [6]. That is, we consider x , the canonical binary (base-2) representation of an integer n , and call N a *binary square* if $N = 0$, or if $x = yy$ for some nonempty string y that starts with a 1. Thus, for example, $N = 221$ is a binary square, since 221 in base 2 is $11011101 = (1101)(1101)$. The first few binary squares are

$$0, 3, 10, 15, 36, 45, 54, 63, 136, 153, 170, 187, 204, 221, 238, 255, \dots;$$

they form sequence [A020330](#) in the *On-Line Encyclopedia of Integer Sequences* [9]. This is a natural sequence to study, since the binary squares have density $\Theta(n^{1/2})$ in the natural numbers, just like the ordinary squares.

In this paper we prove the following result.

Theorem 1. *Every natural number $N > 686$ is the sum of four binary squares. There are 56 exceptions, given below:*

1, 2, 4, 5, 7, 8, 11, 14, 17, 22, 27, 29, 32, 34, 37, 41, 44, 47, 53, 62, 95, 104, 107, 113, 116, 122, 125, 131, 134, 140, 143, 148, 155, 158, 160, 167, 407, 424, 441, 458, 475, 492, 509, 526, 552, 560, 569, 587, 599, 608, 613, 620, 638, 653, 671, 686.

The novelty in our approach is that we obtain this theorem in additive number theory *using very little number theory at all*. Instead, we use an approach based on formal language theory. Previously we obtained similar results for palindromes [11].

Evidently, one could also consider the analogous results for other powers such as cubes, and bases $b \geq 2$, but we do not do that in this paper.

2 The main lemma

We are concerned with the binary representation of numbers, so let us introduce some notation. If N is a natural number, then by $(N)_2$ we mean the string giving the canonical base-2 representation of N , having no leading zeroes. For example, $(43)_2 = 101011$. The canonical representation of 0 is ϵ , the empty string.

If $2^{n-1} \leq N < 2^n$ for $n \geq 1$, we say that N is an n -bit integer in base 2. Note that the first bit of an n -bit integer is always nonzero. The *length* of an integer N satisfying $2^{n-1} \leq N < 2^n$ is defined to be n ; alternatively, the length of N is $1 + \lfloor \log_2 N \rfloor$.

Our main lemma is

Lemma 2.

(a) Every length- n integer, n odd, $n \geq 13$, is the sum of binary squares as follows: either

- one of length $n - 1$ and one of length $n - 3$, or
- two of length $n - 1$ and one of length $n - 3$, or
- one of length $n - 1$ and two of length $n - 3$, or
- one each of lengths $n - 1$, $n - 3$, and $n - 5$, or
- two of length $n - 1$ and two of length $n - 3$, or
- two of length $n - 1$, one of length $n - 3$, and one of length $n - 5$.

(b) Every length- n integer, n even, $n \geq 18$, is the sum of binary squares as follows: either

- two of length $n - 2$ and two of length $n - 4$, or
- three of length $n - 2$ and one of length $n - 4$, or
- one each of lengths n , $n - 4$, and $n - 6$, or
- two of length $n - 2$, one of length $n - 4$, and one of length $n - 6$.

Lemma 2 almost immediately proves Theorem 1:

Proof. If $N < 2^{17} = 131072$, the result can be proved by a straightforward computation. There are

- 256 binary squares $< 2^{17}$;

- 19542 numbers $< 2^{17}$ that are the sum of two binary squares;
- 95422 numbers $< 2^{17}$ that are the sum of three binary squares;
- 131016 numbers $< 2^{17}$ that are the sum of four binary squares.

Otherwise $N \geq 2^{17}$, so $(N)_2$ is a binary string of length $n \geq 18$. If n is odd, the result follows from Lemma 2 (a). If n is even, the result follows from Lemma 2 (b). \square

It now remains to prove Lemma 2. We do this in the next section.

3 Proof of Lemma 2

Proof. The basic idea is to use nondeterministic finite automata (NFAs). These are finite-state machines where each input corresponds to multiple computational paths; an input is accepted iff *some* computational path leads to a final state. We assume the reader is familiar with the basics of this theory; if not, please consult, e.g., [6]. For us, an NFA is a quintuple $(Q, \Sigma, \delta, q_0, F)$, where Q is the set of states, Σ is the input alphabet, δ is the transition function, q_0 is the initial state, and F is the set of final states.

We construct an NFA that, on input an integer N written in binary, “guesses” a representation as a sum of binary squares, and then verifies that the sum is indeed N . Everything is done using a reversed representation, with least significant digits processed first. There are some complications, however.

First, with an NFA we cannot verify that a guessed string is indeed a binary square, as the language $\{xx : x \in 1\{0,1\}^*\}$ is not a regular language. So instead we only guess the “first half” of a binary square. Now, however, we are forced to choose a slightly unusual representation for N , in order to be able to compare the sum of our guessed powers with the input N . If N were represented in its ordinary base-2 representation, this would be impossible with an NFA, since once we process the guessed “first half” and compare it to the input, we would no longer have the “second half” (identical to the first) to compare to the rest of the input.

To get around this problem, we represent integers N in a kind of “folded representation” over the input alphabet $\Sigma_2 \cup (\Sigma_2 \times \Sigma_2)$, where $\Sigma_k = \{0, 1, \dots, k-1\}$. The idea is to present our NFA with two bits of the input string at once, so that we can add both halves of our guessed powers at the same time, verifying that we are producing N as we go. Note that we use slightly different representations for the two parts of Lemma 2. The precise representations are detailed in their respective subsections.

We can now prove Lemma 2 by phrasing it as a language inclusion problem. For each of the two parts of the lemma, we can build an NFA A that only accepts such folded strings if they represent numbers that are the sum of any of the combination of squares as described in the lemma. We also create an NFA, B , that accepts all valid folded representations that are sufficiently long. We then check the assertion that the language recognized by B is a subset of that recognized by A .

3.1 Odd-length inputs

In order to flag certain positions of the input tape, we use an extended alphabet. Define

$$\Gamma = \{1_f\} \cup \bigcup_{\alpha \in \{a,b,c,d,e\}} \{[0,0]_\alpha, [0,1]_\alpha, [1,0]_\alpha, [1,1]_\alpha\}.$$

Let N be an integer, and let $n = 2i + 1$ be the length of its binary representation. We write $(N)_2 = a_{2i}a_{2i-1} \cdots a_1a_0$ and fold this to produce the input string

$$[a_i, a_0]_a [a_{i+1}, a_1]_a \cdots [a_{2i-5}, a_{i-5}]_a [a_{2i-4}, a_{i-4}]_b [a_{2i-3}, a_{i-3}]_c [a_{2i-2}, a_{i-2}]_d [a_{2i-1}, a_{i-1}]_e a_{2i_f}.$$

Let A_{odd} be the NFA that recognizes those odd-length integers, represented in this folded format, that are the sum of binary squares meeting any of the 6 conditions listed in Lemma 2 (a). We construct A_{odd} as the union of several automata $A(t_{n-1}, t_{n-3}, m_a)$ and $B(t_{n-1}, t_{n-3}, t_{n-5}, m_b)$. The parameters t_p represent the number of summands of length p we are guessing. The parameters m_a and m_b are the carries that we are guessing will be produced by the first half of the summed binary squares. A -type machines try summands of lengths $n - 1$ and $n - 3$ only, while B -type machines include at least one $(n - 5)$ -length summand. We note that for the purpose of summing, guessing t binary squares is equivalent to guessing a *single* square over the larger alphabet Σ_{t+1} .

We now consider the construction of a single automaton

$$A(t_{n-1}, t_{n-3}, m) = (Q \cup \{q_{\text{acc}}, q_0, s_1\}, \Gamma, \delta, q_0, \{q_{\text{acc}}\}).$$

The elements of Q have 4 non-negative parameters and are of the form $q(x_1, x_2, c_1, c_2)$. The parameter $x_1 \leq t_{n-3}$ is the last digit of the guessed summand of length $n - 3$ and $x_2 \leq t_{n-3}$ is the previous higher guess of the length- $n - 3$ summand; this must be the next lower guess of this summand. We use c_1 to track the higher carry, and c_2 to track the lower carry. We must have $c_1, c_2 < t_{n-1} + t_{n-3}$.

We now discuss the transition function, δ of our NFA. In this section, we say that the sum of natural numbers, μ_1 and μ_2 , “produces” an output bit of $\theta \in \Sigma_2$ with a “carry” of γ if $\mu_1 + \mu_2 \equiv \theta \pmod{2}$ and $\gamma = \lfloor \mu_1 + \mu_2 \rfloor$.

We allow a transition from q_0 to $q(x_1, x_2, c_1, c_2)$ on the letter $[j, k]_a$ iff there exists $0 \leq r \leq t_{n-1}$ such that $x_2 + r + m$ produces an output of j with a carry of c_1 and $x_1 + r$ produces an output of k with a carry of c_2 .

We allow a transition from $q(x_1, x_2, c_1, c_2)$ to $q(x'_1, x'_2, c'_1, c'_2)$ on the letters $[j, k]_a$ and $[j, k]_b$ iff there exists $0 \leq r \leq t_{n-1}$ such that $x'_2 + r + c_1$ produces an output of j with a carry of c'_1 and $x_2 + r + c_2$ produces an output of k with a carry of c'_2 . Elements of Q have identical transitions on inputs with subscripts a and b . The reason we have the letters with subscript b is for B -machines, which guess a summand of length $n - 5$.

There is only one letter of the input with the subscript c , and it corresponds to the last higher guess of the summand of length $n - 3$. We allow a transition from $q(x_1, x_2, c_1, c_2)$ to $q(x'_1, t_{n-3}, c'_1, c'_2)$ on the letter $[j, k]_c$ iff there exists $0 \leq r \leq t_{n-1}$ such that $t_{n-3} + r + c_1$

produces an output of j with a carry of c'_1 and $x_2 + r + c_2$ produces an output of k with a carry of c'_2 .

There is only one letter of the input with the subscript d , and it corresponds to the second-last lower guess of the summand of length $n - 3$. We allow a transition from $q(x_1, t_{n-3}, c_1, c_2)$ to $q(x'_1, 0, c'_1, c'_2)$ on the letter $[j, k]_d$ iff there exists $0 \leq r \leq t_{n-1}$ such that $r + c_1$ produces an output of j with a carry of c'_1 and $t_{n-3} + r + c_2$ produces an output of k with a carry of c'_2 .

There is only one letter of the input with the subscript e , and it corresponds to the last lower guess of the summand of length $n - 3$. We allow a transition from $q(x_1, 0, c_1, c_2)$ to s_1 on the letter $[j, k]_e$ iff $t_{n-1} + c_1$ produces an output of j with a carry of 1 and $x_1 + t_{n-1} + c_2$ produces an output of k with a carry of m .

Finally, we add a transition from s_1 to q_{acc} on the letter 1_f .

We now consider the construction of a single automaton

$$B(t_{n-1}, t_{n-3}, t_{n-5}, m) = (P \cup Q \cup \{q_{\text{acc}}, q_0, s_1\}, \Gamma, \delta, q_0, \{q_{\text{acc}}\}).$$

The elements of P have 6 non-negative parameters and are of the form

$$q(x_1, x_2, y_1, y_3, c_1, c_2).$$

The parameter $x_1 \leq t_{n-3}$ is the last digit of the guessed summand of length $n - 3$ and $x_2 \leq t_{n-3}$ is the previous higher guess of the length- $n - 3$ summand. The parameter $y_1 \leq t_{n-5}$ is the last digit of the guessed summand of length $n - 5$ and $y_3 \leq t_{n-5}$ is the previous higher guess of the length- $n - 5$ summand. We use c_1 to track the higher carry, and c_2 to track the lower carry. We must have $c_1, c_2 < t_{n-1} + t_{n-3} + t_{n-5}$.

The elements of Q have 8 non-negative parameters and are of the form

$$q(x_1, x_2, y_1, y_2, y_3, y_4, c_1, c_2).$$

The parameter $x_1 \leq t_{n-3}$ is the last digit of the guessed summand of length $n - 3$ and $x_2 \leq t_{n-3}$ is the previous higher guess of the length- $n - 3$ summand. The parameters $y_1, y_2 \leq t_{n-5}$ are the last digit and the second-last digit of the guessed summand of length $n - 5$ respectively. The parameter $y_3, y_4 \leq t_{n-5}$ are the two most recent higher guess of the length- $n - 5$ summand, with y_4 being the most recent one. We use c_1 to track the higher carry, and c_2 to track the lower carry. We must have $c_1, c_2 < t_{n-1} + t_{n-3} + t_{n-5}$.

We now discuss the transition function, δ of our NFA. We allow a transition from q_0 to $p(x_1, x_2, y_1, y_3, c_1, c_2)$ on the letter $[j, k]_a$ iff there exists $0 \leq r \leq t_{n-1}$ such that $x_2 + y_3 + r + m$ produces an output of j with a carry of c_1 and $x_1 + y_1 + r$ produces an output of k with a carry of c_2 .

We use a transition from $p(x_1, x_2, y_1, y_3, c_1, c_2)$ to $q(x_1, x'_2, y_1, y'_2, y_3, y'_4, c'_1, c'_2)$ on the letter $[j, k]_a$ iff there exists $0 \leq r \leq t_{n-1}$ such that $x'_2 + y'_4 + r + c_1$ produces an output of j with a carry of c_1 and $x_2 + y'_2 + r + c_2$ produces an output of k with a carry of c_2 .

We use a transition from $q(x_1, x_2, y_1, y_2, y_3, y_4, c_1, c_2)$ to $q(x_1, x'_2, y_1, y_2, y_4, y'_4, c'_1, c'_2)$ on the letter $[j, k]_a$ iff there exists $0 \leq r \leq t_{n-1}$ such that $x'_2 + y'_4 + r + c_1$ produces an output of j with a carry of c_1 and $x_2 + y_3 + r + c_2$ produces an output of k with a carry of c_2 .

We use a transition from $q(x_1, x_2, y_1, y_2, y_3, t_{n-5}, c_1, c_2)$ to $q(x_1, x'_2, y_1, y_2, t_{n-5}, t_{n-5}, c'_1, c'_2)$ on the letter $[j, k]_b$ iff there exists $0 \leq r \leq t_{n-1}$ such that $x'_2 + r + c_1$ produces an output of j with a carry of c_1 and $x_2 + y_3 + r + c_2$ produces an output of k with a carry of c_2 .

We use a transition from $q(x_1, x_2, y_1, y_2, t_{n-5}, t_{n-5}, c_1, c_2)$ to $q(x_1, t_{n-3}, y_1, y_2, t_{n-5}, t_{n-5}, c'_1, c'_2)$ on the letter $[j, k]_c$ iff there exists $0 \leq r \leq t_{n-1}$ such that $t_{n-3} + r + c_1$ produces an output of j with a carry of c_1 and $x_2 + y_3 + r + c_2$ produces an output of k with a carry of c_2 .

We use a transition from $q(x_1, t_{n-3}, y_1, y_2, t_{n-5}, t_{n-5}, c_1, c_2)$ to $q(x_1, t_{n-3}, y_1, y_2, t_{n-5}, t_{n-5}, c'_1, c'_2)$ on the letter $[j, k]_d$ iff there exists $0 \leq r \leq t_{n-1}$ such that $r + c_1$ produces an output of j with a carry of c_1 and $t_{n-3} + y_1 + r + c_2$ produces an output of k with a carry of c_2 .

We use a transition from $q(x_1, t_{n-3}, y_1, y_2, t_{n-5}, t_{n-5}, c_1, c_2)$ to s_1 on the letter $[j, k]_e$ iff $t_{n-1} + c_1$ produces an output of j with a carry of 1 and $x_1 + y_2 + t_{n-1} + c_2$ produces an output of k with a carry of m .

Finally, we add a transition from s_1 to q_{acc} on the letter 1_f .

We now turn to verification of the inclusion assertion. We used the Automata Library toolchain of the ULTIMATE program analysis framework [4, 5] to establish our results. The ULTIMATE code proving our result can be found in the file `OddSquareConjecture.ats` at <https://cs.uwaterloo.ca/~shallit/papers.html>. Since the constructed machines get very large, we wrote a C++ program generating these machines, which can be found in the file `OddSquares.cpp` at <https://cs.uwaterloo.ca/~shallit/papers.html>.

The final machine, A_{odd} , has 2258 states. The syntax checker, B , has 8 states. We then asserted that the language recognized by B is a subset of that recognized by A . ULTIMATE verified this assertion in under a minute. Since this test succeeded, the proof of Lemma 2 (a) is complete.

3.2 Even-length inputs

In order to flag certain positions of the input tape, we use an extended alphabet. Define

$$\Gamma = \left(\bigcup_{\alpha \in \{a, b, c, d, e\}} \{[0, 0]_\alpha, [0, 1]_\alpha, [1, 0]_\alpha, [1, 1]_\alpha\} \right) \cup \left(\bigcup_{\beta \in \{f, g, h, i\}} \{0_\beta, 1_\beta\} \right).$$

Let N be an integer, and let $n = 2i + 4$ be the length of its binary representation. We write $(N)_2 = a_{2i+3}a_{2i+2} \cdots a_1a_0$ and fold this to produce the input string

$$[a_i, a_0]_a [a_{i+1}, a_1]_b [a_{i+2}, a_2]_c [a_{i+3}, a_3]_c \cdots [a_{2i-3}, a_{i-3}]_c [a_{2i-2}, a_{i-2}]_d [a_{2i-1}, a_{i-1}]_e a_{2i} a_{2i+1} a_{2i+2} a_{2i+3} \cdots$$

Let A_{even} be the NFA that recognizes the even-length integers, represented in this folded format, iff the integer is the sum of binary squares meeting any of the 4 conditions listed in Lemma 2 (b). We construct A_{even} as the union of several automata $A(t_n, t_{n-2}, t_{n-4}, t_{n-6}, m)$. The parameters t_p represent the number of summands of length p we are guessing. The parameter m is the carry that we are guessing will be produced by the first half of the

summed binary squares. We note that for the purpose of summing, guessing t binary squares is equivalent to guessing a *single* square over the larger alphabet Σ_{t+1} .

We now consider the construction of a single automaton

$$A(t_n, t_{n-2}, t_{n-4}, t_{n-6}, m) = (Q \cup \{q_{\text{acc}}\}, \Gamma, \delta, q_0, \{q_{\text{acc}}\}).$$

The elements of Q have 8 non-negative parameters and are of the form

$$q(x_1, x_2, x_3, y_1, z_1, z_2, c_1, c_2).$$

The parameter x_1 is the second digit of the guessed summand of length n . The parameters x_2 and x_3 represent the previous 2 lower guesses of the length- n summand; these must be the next 2 higher guesses of this summand. The parameter y_1 represents the previous lower guess of the length- $(n-2)$ summand. We set z_1 as the last digit of the guessed summand of length $n-6$, while z_2 is the previous higher guess of this summand. Finally, c_1 tracks the lower carry, while c_2 tracks the higher carry. For any p , we must have $x_p \leq t_n$, $y_p \leq t_{n-2}$, $z_p \leq t_{n-6}$, and $c_p < t_n + t_{n-2} + t_{n-4} + t_{n-6}$. The initial state, q_0 , is $q(0, 0, 0, 0, 0, 0, 0, 0)$.

We now discuss the transition function, δ of our NFA. Note that in our representation of even-length integers, the first letter of the input must have the subscript a , and it is the only letter to do so. We only allow the initial state to have outgoing transitions on such letters.

We allow a transition from q_0 to $q(x_1, 0, x_3, y_1, z_1, z_2, c_1, c_2)$ on the letter $[j, k]_a$ iff there exists $0 \leq r \leq t_{n-4}$ such that $x_1 + t_{n-2} + r + z_2 + m$ produces an output of j with a carry of c_2 and $x_3 + y_1 + r + z_1$ produces an output of k with a carry of c_1 .

The second letter of the input must have the subscript b , and it is the only letter to do so. We allow a transition from $q(x_1, 0, x_3, y_1, z_1, z_2, c_1, c_2)$ to $q(x_1, x_3, x'_3, y'_1, z_1, z'_2, c'_1, c'_2)$ on the letter $[j, k]_b$ iff there exists $0 \leq r \leq t_{n-4}$ such that $t_n + y_1 + r + z'_2 + c_2$ produces an output of j with a carry of c'_2 and $x'_3 + y'_1 + r + z_2 + c_1$ produces an output of k with a carry of c'_1 .

We allow a transition from $q(x_1, x_2, x_3, y_1, z_1, z_2, c_1, c_2)$ to $q(x_1, x_3, x'_3, y'_1, z_1, z'_2, c'_1, c'_2)$ on the letter $[j, k]_c$ iff there exists $0 \leq r \leq t_{n-4}$ such that $x_2 + y_1 + r + z'_2 + c_2$ produces an output of j with a carry of c'_2 and $x'_3 + y'_1 + r + z_2 + c_1$ produces an output of k with a carry of c'_1 .

The letter of the input with the subscript d corresponds to the last guess of the lower half of the summand of length $n-6$, and it is the only letter to do so. We allow a transition from $q(x_1, x_2, x_3, y_1, z_1, t_{n-6}, c_1, c_2)$ to $q(x_1, x_3, x'_3, y'_1, z_1, 0, c'_1, c'_2)$ on the letter $[j, k]_d$ iff there exists $0 \leq r \leq t_{n-4}$ such that $x_2 + y_1 + r + c_2$ produces an output of j with a carry of c'_2 and $x'_3 + y'_1 + r + t_{n-6} + c_1$ produces an output of k with a carry of c'_1 .

The letter of the input with the subscript e corresponds to the last guess of both halves of the summand of length $n-4$, and it is the only letter to do so. We allow a transition from $q(x_1, x_2, x_3, y_1, z_1, 0, c_1, c_2)$ to $q(x_1, x_3, x'_3, y'_1, 0, 0, 0, c'_2)$ on the letter $[j, k]_e$ iff $x_2 + y_1 + t_{n-4} + c_2$ produces an output of j with a carry of c'_2 and $x'_3 + y'_1 + t_{n-4} + z_1 + c_1$ produces an output of k with a carry of m .

We allow a transition from $q(x_1, x_2, x_3, y_1, 0, 0, 0, c_2)$ to $q(x_1, x_3, 0, 0, 0, 0, 0, c'_2)$ on the letter j_f iff $x_2 + y_1 + c_2$ produces an output of j with a carry of c'_2 .

We allow a transition from $q(x_1, x_2, 0, 0, 0, 0, c_2)$ to $q(x_1, 0, 0, 0, 0, 0, c'_2)$ on the letter j_g iff $x_2 + t_{n-2} + c_2$ produces an output of j with a carry of c'_2 .

We allow a transition from $q(x_1, 0, 0, 0, 0, 0, c_2)$ to $q(0, 0, 0, 0, 0, 0, c'_2)$ on the letter j_h iff $x_1 + c_2$ produces an output of j with a carry of c'_2 .

We allow a transition from $q(0, 0, 0, 0, 0, 0, c_2)$ to q_{acc} on the letter 1_i iff $t_n + c_2$ produces an output of 1 with a carry of 0.

The final machine, A_{even} is constructed as the union of 15 automata:

- $A(0, 2, 2, 0, m)$, varying m from 0 to 3
- $A(0, 3, 1, 0, m)$, varying m from 0 to 3
- $A(1, 0, 1, 1, m)$, varying m from 0 to 2
- $A(0, 2, 1, 1, m)$, varying m from 0 to 3

We now turn to verification of the inclusion assertion. The `ULTIMATE` code proving our result can be found in the file `EvenSquareConjecture.ats` at <https://cs.uwaterloo.ca/~shallit/papers>. Since the constructed machines get very large, we wrote a C++ program generating these machines, which can be found in the file `EvenSquares.cpp` at <https://cs.uwaterloo.ca/~shallit/papers>.

The final machine, A_{even} , has 1343 states. The syntax checker, B , has 12 states. We then asserted that the language recognized by B is a subset of that recognized by A . `ULTIMATE` verified this assertion in under a minute. Since this test succeeded, the proof of Lemma 2 (b) is complete. □

4 Other results

Our technique can be used to obtain other results in additive number theory. For example, recently Crocker [1] and Platt & Trudgian [10] studied the integers representable as the sum of two ordinary squares and two powers of 2.

Lemma 3.

(a) *Every length- n integer, n odd, $n \geq 7$, is the sum of at most two powers of 2 and either:*

- *at most two squares of length $n - 1$, or*
- *at most one square of length $n - 1$ and one of length $n - 3$.*

(b) *Every length- n integer, n even, $n \geq 10$, is the sum of at most two powers of 2 and either:*

- *at most one square of length n and one of length $n - 4$, or*
- *at most one square of length $n - 2$ and one of length $n - 4$.*

Proof. We use a similar proof strategy as before. The `ULTIMATE` code proving our result can be found in the files `OddSquarePowerConjecture.ats` and `EvenSquarePowerConjecture.ats` at <https://cs.uwaterloo.ca/~shallit/papers.html>. The generators can be found as `OddSquarePower.cpp` and `EvenSquarePower.cpp` at <https://cs.uwaterloo.ca/~shallit/papers.html>.

The final machines for the odd-length and even-length cases have 806 and 2175 states respectively. The language inclusion assertions all hold. This concludes the proof. \square

We thus have the following theorem:

Theorem 4. *Every natural number N is the sum of at most two binary squares and at most two powers of 2.*

Proof. For $N < 512$, the result can be easily verified. Otherwise, we use Lemma 3 (a) if N is an odd-length binary number and Lemma 3 (b) if it is even. \square

We also consider the notion of *generalized binary squares*. A number N is called a generalized binary square if one can concatenate 0 or more leading zeroes to its binary representation to produce a binary square. As an example, 9 is a generalized square, since 9 in base 2 is 1001, which can be written as $001001 = (001)(001)$. The first few generalized binary squares are

$$0, 3, 5, 9, 10, 15, 17, 18, 27, 33, 34, 36, 45, 51, 54, 63, \dots;$$

they form sequence [A175468](#) in the *On-Line Encyclopedia of Integer Sequences* [9].

In what follows, when we refer to the length of a generalized square, we mean the length including the leading zeroes. Thus, 9 is a generalized square of length 6 (and not 4).

Lemma 5.

- (a) *Every length- n integer, $n \geq 7$, n odd, is the sum of 3 generalized squares, of lengths $n + 1$, $n - 1$, and $n - 3$.*
- (b) *Every length- n integer, $n \geq 8$, n even, is the sum of 3 generalized squares, of lengths n , $n - 2$, and $n - 4$.*

Proof. We use a very similar proof strategy as in the proof of Lemma 2. We drop the requirement that the most significant digit of our guessed squares be 1, thus allowing for generalized squares. Note that the square of length $n + 1$ must in part (a) must start with a 0.

The `ULTIMATE` code proving our result can be found in the files `OddGenSquareConjecture.ats` and `EvenGenSquareConjecture.ats` at <https://cs.uwaterloo.ca/~shallit/papers.html>. The generators can be found as `OddGeneralizedSquares.cpp` and `EvenGeneralizedSquares.cpp` at <https://cs.uwaterloo.ca/~shallit/papers.html>. The final machines for the odd-length and even-length cases have 132 and 263 states respectively. \square

We thus have the following theorem:

Theorem 6. *Every natural number $N > 7$ is the sum of 3 generalized squares.*

Proof. For $7 < N < 64$ the result can be easily verified. Otherwise, we use Lemma 5 (a) is an odd-length binary number and Lemma 5 (b) if it is even. \square

5 Further work

We do not currently know whether the number 4 in Theorem 1 is optimal, although numerical evidence strongly suggests that it is.

Numerical evidence suggests the following two conjectures:

Conjecture 7. Let α_3 denote the asymptotic density of the set S_3 of natural numbers that are the sum of three binary squares. Then $\alpha_3 < 0.9$.

We could also focus on sums of *positive* binary squares. (For the analogous problem dealing with ordinary squares, see, e.g., [3, Chapter 6].) It seems likely that our method could be used to prove the following result.

Conjecture 8. Every natural number > 1772 is the sum of exactly four positive binary squares. There are 112 exceptions, given below:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 25, 27, 28, 29, 30, 32, 34, 35, 37, 39, 41, 42, 44, 46, 47, 49, 51, 53, 56, 58, 62, 65, 67, 74, 83, 88, 95, 100, 104, 107, 109, 113, 116, 122, 125, 131, 134, 140, 143, 148, 149, 155, 158, 160, 161, 167, 170, 173, 175, 182, 184, 368, 385, 402, 407, 419, 424, 436, 441, 458, 475, 492, 509, 526, 543, 552, 560, 569, 587, 599, 608, 613, 620, 625, 638, 647, 653, 671, 686, 698, 713, 1508, 1541, 1574, 1607, 1640, 1673, 1706, 1739, 1772.

References

- [1] R. C. Crocker. On the sum of two squares and two powers of k . *Colloq. Math.* **112** (2008), 235–267.
- [2] W. J. Ellison. Waring’s problem. *Amer. Math. Monthly* **78** (1971), 10–36.
- [3] E. Grosswald. *Representations of Integers as Sums of Squares*. Springer-Verlag, 1985.
- [4] M. Heizmann, J. Hoenicke, and A. Podelski. Software model checking for people who love automata. In N. Sharygina and H. Veith, editors, *Computer Aided Verification — 25th International Conference, CAV 2013*, Vol. 8044 of *Lecture Notes in Computer Science*, pp. 36–52. Springer-Verlag, 2013.

- [5] M. Heizmann, D. Dietsch, M. Greitschus, J. Leike, B. Musa, C. Schätzle, and A. Podelski. Ultimate automizer with two-track proofs. In M. Chechik and J.-F. Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems — 22nd International Conference, TACAS 2016*, Vol. 9636 of *Lecture Notes in Computer Science*, pp. 950–953. Springer-Verlag, 2016.
- [6] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [7] C. J. Moreno and S. S. Wagstaff, Jr. *Sums of Squares of Integers*. Chapman and Hall/CRC, 2005.
- [8] M. B. Nathanson. *Additive Number Theory: The Classical Bases*. Springer-Verlag, 1996.
- [9] N. J. A. Sloane. The on-line encyclopedia of integer sequences. Available at <https://oeis.org>, 2016.
- [10] D. Platt and T. Trudgian. On the sum of two squares and at most two powers of 2. Preprint, available at <https://arxiv.org/abs/1610.01672>, 2016.
- [11] A. Rajasekaran, J. Shallit, and T. Smith. Sums of palindromes: an approach via nested-word automata. Preprint, available at <https://arxiv.org/abs/1706.10206>, 2017.
- [12] C. Small. Waring’s problem. *Math. Mag.* **50** (1977), 12–16.
- [13] R. C. Vaughan and T. Wooley. Waring’s problem: a survey. In M. A. Bennett, B. C. Berndt, N. Boston, H. G. Diamond, A. J. Hildebrand, and W. Philipp, editors, *Number Theory for the Millennium. III*, pp. 301–340. A. K. Peters, 2002.