Theory of Computing Systems
https://doi.org/10.1007/s00224-019-09929-9

Additive Number Theory via Automata Theory

Aayush Rajasekaran[1] · Jeffrey Shallit[1] · Tim Smith[1]

Published online: 29 May 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

We show how some problems in additive number theory can be attacked in a novel way, using techniques from the theory of finite automata. We start by recalling the relationship between first-order logic and finite automata, and use this relationship to solve several problems involving sums of numbers defined by their base-2 and Fibonacci representations. Next, we turn to harder results. Recently, Cilleruelo, Luca, & Baxter proved, for all bases $b \geq 5$, that every natural number is the sum of at most 3 natural numbers whose base-b representation is a palindrome (Cilleruelo et al., Math. Comput. **87**, 3023–3055, 2018). However, the cases $b = 2, 3, 4$ were left unresolved. We prove that every natural number is the sum of at most 4 natural numbers whose base-2 representation is a palindrome. Here the constant 4 is optimal. We obtain similar results for bases 3 and 4, thus completely resolving the problem of palindromes as an additive basis. We consider some other variations on this problem, and prove similar results. We argue that heavily case-based proofs are a good signal that a decision procedure may help to automate the proof.

Keywords Additive number theory · Formal language theory · Automata theory · Visibly pushdown automaton · Palindrome · Decision procedure · Automated proof

This article is part of the Topical Collection on *Special Issue on Theoretical Aspects of Computer Science (2018)*

✉ Jeffrey Shallit
shallit@uwaterloo.ca

Aayush Rajasekaran
arajasekaran@uwaterloo.ca

Tim Smith
timsmith@uwaterloo.ca

[1] School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada

1 Introduction

In this paper we combine three different themes: (i) additive number theory; (ii) numbers with special kinds of representations in base k; (iii) use of a decision procedure to prove theorems in an automated fashion. Our main result (Corollary 9) is that every natural number is the sum of at most 4 numbers whose base-2 representation is a palindrome.

Additive number theory is the study of the additive properties of integers. For example, Lagrange proved (1770) that every natural number is the sum of four squares [40]. In additive number theory, a subset $S \subseteq \mathbb{N}$ is called a *additive basis of order h* if every element of $\mathbb{N}$ can be written as a sum of at most h members of S, not necessarily distinct.

Waring's problem asks for the smallest value $g(k)$ such that the k'th powers form a basis of order $g(k)$. In a variation on Waring's problem, one can ask for the smallest value $G(k)$ such that every *sufficiently large* natural number is the sum of $G(k)$ k'th powers [77]. This kind of representation is called an *asymptotic additive basis* of order $G(k)$.

Quoting Nathanson [61, p. 7],

> *"The central problem in additive number theory is to determine if a given set of integers is a basis of finite order."*

In this paper we show how to solve this central problem for certain sets, *using almost no number theory at all*.

Our second theme concerns numbers with special representations in base k. For example, numbers of the form $11 \cdots 1$ in base k are sometimes called *repunits* [78], and special effort has been devoted to factoring such numbers, with the Mersenne numbers $2^n - 1$ being the most famous examples. The *Nagell-Ljunggren problem* asks for a characterization of those repunits that are integer powers (see, e.g., [70]).

Another interesting class consists of those numbers whose base-k representation forms a *palindrome*: a word that reads the same forwards and backwards, like the English word `radar`. Palindromic numbers have been studied for some time in number theory; see, for example, [7, 12, 47, 49, 50, 72, 75] for squares whose base-k representation is a palindrome; [29, 41] for palindromes in arithmetic progressions; [28, 53, 54] for palindromes in recurrence sequences; [9, 10] for palindromic aspects of prime and composite numbers; [27, 44] for palindromes that are powers; and [2, 11, 13, 14, 17, 37, 48, 55, 65, 77] for various other aspects.

Recently Banks initiated the study of the additive properties of palindromes, proving that every natural number is the sum of at most 49 numbers whose decimal representation is a palindrome [8]. (Also see [71].) Banks' result was improved by Cilleruelo, Luca, & Baxter [25, 26], who proved that for all bases $b \geq 5$, every natural number is the sum of at most 3 numbers whose base-b representation is a palindrome. The proofs of Banks and Cilleruelo, Luca, & Baxter are both rather lengthy and case-based. Up to now, there have been no results proved for bases $b = 2, 3, 4$.

The long case-based solutions to the problem of representation by sums of palindromes suggests that perhaps a more automated approach might be useful. For example, in a series of recent papers, the second author and his co-authors have proved a number of old and new results in combinatorics on words using a decision procedure based on first-order logic [1, 24, 30, 31, 33–36, 60, 69]. The classic result of Thue [19, 74] that the Thue-Morse infinite word $\mathbf{t} = 0110100110010110\cdots$ avoids overlaps (that is, words of the form $axaxa$ where a is a single letter and x is a possibly empty word) is an example of a case-based proof that can be entirely replaced [1] with a decision procedure based on the first-order logical theory $\mathrm{FO}(\mathbb{N}, +, V_2)$, where V_k is the function mapping n to the largest power of k dividing n.

Inspired by these and other successes in automated deduction and theorem-proving (e.g., [57]), we turn to formal languages and automata theory as a suitable framework for expressing the palindrome representation problem. Since we want to make assertions about the representations of *all* natural numbers, this requires finding (a) a machine model or logical theory in which universality is decidable and (b) a variant of the additive problem of palindromes suitable for this machine model or logical theory. The first model we use is the *nested-word automaton*, a variant of the more familiar pushdown automaton. This is used to handle the case for base $b = 2$. The second model we use is the ordinary finite automaton. We use this model to resolve the cases $b = 3, 4$.

Our paper is organized as follows: In Section 2 we introduce some notation and terminology. In Section 3 we describe how first-order logic and automata can be used to solve problems in additive number theory. In Section 4 we introduce additive problems on palindromes, and state more precisely the problem we want to solve. In Section 5 we recall the pushdown automaton model and give an example, and we motivate our use of nested-word automata. In Section 6 we restate our problem in the framework of nested-word automata, and the proof of a bound of 4 palindromes in the binary case is given in Section 7. The novelty of our approach involves replacing the long case-based reasoning of previous proofs with an automaton-based approach using a decision procedure. Section 8 provides our results for bases 3 and 4. In Section 9 we consider some variations on the original problem. In Section 10 we discuss possible objections to our approach. In Section 11 we mention some open problems. Finally, in Section 12, we conclude our paper by stating a thesis underlying our approach.

This is an extended version of a paper that was presented at STACS 2018 [67]. It also forms part of the master's thesis of the first author [66]. For more results along these lines, see [15, 16, 46, 56].

2 Notation

We introduce some notation and terminology. The natural numbers are $\mathbb{N} = \{0, 1, 2, \ldots\}$. If n is a natural number, then by $(n)_k$ we mean the word (i.e., the string of symbols) representing n in base k, with no leading zeroes, starting with the most significant digit. Thus, for example, $(43)_2 = 101011$. This notation is extended to sets of integers in the obvious way: $(S)_k = \{(n)_k : n \in S\}$. The alphabet Σ_k is defined to be $\{0, 1, \ldots, k-1\}$; by Σ_k^* we mean the set of all finite words over Σ_k. If $x \in \Sigma_\ell^*$ for some ℓ, then by $[x]_k$ we mean the integer represented by the word x, considered as if it were a number in base k, with the most significant digit at the left. That is, if $x = a_1 a_2 \cdots a_n$, then $[x]_k = \sum_{1 \le i \le n} a_i k^{n-i}$. For example, $[135]_2 = 15$.

There is another numeration system, based on Fibonacci numbers, that will concern us briefly in this paper: it is the so-called Fibonacci or Zeckendorf system [52, 79]. In this system, integers are represented as a sum of Fibonacci numbers

$$\sum_{0 \le i \le t} e_i F_{i+2}, \tag{1}$$

where $e_i \in \{0, 1\}$ and $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$.

It is well-known that, provided one imposes the condition $e_i e_{i+1} \neq 1$ (that is, the representation contains no two consecutive 1's), its Fibonacci representation is unique. Given an integer $n \ge 0$, we let $(n)_F$ denote this canonical Fibonacci representation $e_t e_{t-1} \cdots e_0$ having no two consecutive 1's. Given a word $e_t e_{t-1} \cdots e_0$ over Σ_2, we let $[e_t e_{t-1} \cdots e_0]_F$ denote the value of the sum (1).

If x is a word, then x^i denotes the word $\overbrace{xx \cdots x}^{i}$, and x^R denotes the reverse of x. Thus, for example, $(\texttt{ma})^2 = \texttt{mama}$, and $(\texttt{drawer})^R = \texttt{reward}$. If $x = x^R$, then x is said to be a *palindrome*. Examples of palindromes in English include `kayak` and `redder`.

3 The First-order Logic Connection

We know, from the pioneering work of Büchi [23] on the one hand, and Bruyère, Hansel, Michaux, and Villemaire [22] on the other, that there is a deep connection between finite automata and first-order logic. We can exploit this connection to obtain results about the additive properties of certain subsets.

Call a subset $S \subseteq \mathbb{N}$ *k-automatic* if there exists a deterministic finite automaton (DFA) M that recognizes the language $(S)_k$ of base-k representations of elements of S. Then we have the following result [22].

Theorem 1 *Fix a k-automatic set S. There is an algorithm that, given a first-order statement φ over the natural numbers, using the operations of membership queries in S, addition, equality, logical operations* `AND`, `OR`, `NOT`*, and the quantifiers $\forall$ and $\exists$, constructs a finite automaton accepting exactly the base-k representations of the values of the free variables in φ that make φ true. If φ has no free variables, the algorithm returns "true" or "false".*

This algorithm has been implemented by Hamoon Mousavi, in a free software program called `Walnut`, available at https://github.com/hamousavi/Walnut and discussed in [59]. We can use it to do additive number theory. In this section we briefly give some examples of what can be proven using this idea. For other results along these lines, see [15, 16].

We start with some results about the so-called "evil" and "odious" numbers. A number is *evil* if the number of 1's in its base-2 representation is even. Otherwise, if the number of 1's is odd, it is *odious*. (We are not responsible for this terminology, which originates in [18, p. 431].) The evil numbers

$$0, 3, 5, 6, 9, 10, 12, 15, 17, 18, 20, 23, 24, 27, 29, 30, \ldots$$

form sequence A001969 and the odious numbers

$$1, 2, 4, 7, 8, 11, 13, 14, 16, 19, 21, 22, 25, 26, 28, 31, \ldots$$

form sequence A000069 in the On-Line Encyclopedia of Integer Sequences (OEIS) [73]. Using Theorem 1 and `Walnut`, we can investigate the additive properties of these numbers. To do so, we recast the definitions in terms of the classical Thue-Morse sequence

$$\mathbf{t} = t_0 t_1 t_2 \cdots = 0110100110010110 \cdots .$$

Here t_n is the sum (mod 2) of the bits in the base-2 representation of n. Thus n is evil if $t_n = 0$ and odious if $t_n = 1$. With this characterization, we easily see that both the evil and odious numbers are 2-automatic.

Theorem 2 *(a) A natural number is the sum of exactly two evil numbers iff it is neither 2, 4, nor of the form* $2 \cdot 4^i - 1$ *for* $i \geq 0$.
(b) A natural number is the sum of exactly two odious numbers iff it is not of the form $2 \cdot 4^i - 1$ *for* $i \geq 0$.

Proof Let us write a first-order statement to assert that n is the sum of two evil numbers. It is

$$\exists i, j\ (n = i + j) \wedge (t_i = 0) \wedge (t_j = 0).$$

We can now translate this statement into `Walnut`. The translation is more or less straightforward:

```
eval sum2e "E i,j (n=i+j) & (T[i]=@0) & (T[j]=@0)":
```

Here `eval` is the `Walnut` command to evaluate a logical expression, `sum2e` is the name assigned to the particular evaluation, the capital `E` is an abbreviation for the quantifier ∃, "there exists", `&` is logical AND, and `T` is a built-in abbreviation for the characteristic sequence of the odious numbers (that is, the Thue-Morse sequence). The expression `@0` represents the constant 0.

When we enter this command in `Walnut`, it produces the automaton in Fig. 1, accepting the binary representations of the free variables that make the logical expression evaluate to true.

A brief inspection shows the automaton rejects exactly the integers specified in the conclusion of part (a). (Note that the base-2 expansion of $2 \cdot 4^i - 1$ is of the form $(11)^i 1$.

The proof for odious numbers is similar, and is left to the reader. □

However, the evil and odious numbers both form an asymptotic additive basis of order 3, as the following theorem shows. We can even demand that the summands be distinct.

Theorem 3 *(a) Every natural number* $N > 10$ *is the sum of 3 distinct evil numbers.*

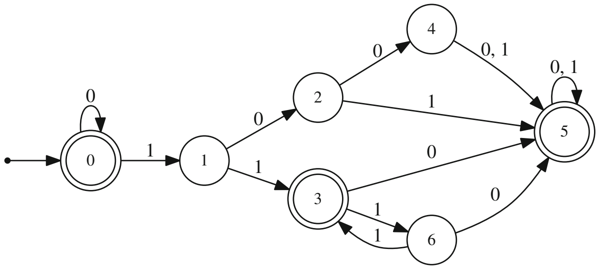

Fig. 1 Automaton for sum of two evil numbers

(b) *Every natural number* $N > 15$ *is the sum of 3 distinct odious numbers.*

Proof Let us write predicates for (a) and (b):

$$\exists i, j, k\ (i \neq j) \wedge (i \neq k) \wedge (j \neq k) \wedge (n = i + j + k)$$
$$\wedge\ (t_i = 0) \wedge (t_j = 0) \wedge (t_k = 0)$$
$$\exists i, j, k\ (i \neq j) \wedge (i \neq k) \wedge (j \neq k) \wedge (n = i + j + k)$$
$$\wedge\ (t_i = 1) \wedge (t_j = 1) \wedge (t_k = 1).$$

These can be translated into `Walnut` as follows:

```
eval sum3de "E i,j,k (i != j) & (i != k) & (j != k) & (n=i+j+k)
  & (T[i] = @0) & (T[j] = @0) & (T[k] = @0)":

eval sum3do "E i,j,k (i != j) & (i != k) & (j != k) & (n=i+j+k)
  & (T[i] = @1) & (T[j] = @1) & (T[k] = @1)":
```

Checking the automata that result from these commands then proves the theorem. □

We can also prove theorems about the *difference* of two evil numbers, or two odious numbers. Consider the following two lemmas, which were recently proved using a case-based argument in a paper about discriminators [39]:

Theorem 4 *(a)* *Let* $i \geq 3$ *and* $1 \leq m < 2^i - 3$. *Then there exist two evil numbers* j, ℓ *with* $0 \leq j < \ell \leq 2^i + 1$ *such that* $m = \ell - j$.

(b) *Let* $i \geq 1$ *and* $1 \leq m < 2^i$. *Then there exist two odious numbers* j, ℓ *with* $1 \leq j < \ell \leq 2^i$ *such that* $m = \ell - j$.

We can construct alternative proofs of these facts using `Walnut`.

Proof First we need a predicate asserting that a number is a power of 2. In first-order logic, we can write, for example, $V_2(k) = k$, where V_2 is the function of Section 1. To translate this into Walnut, we use a regular expression:

```
reg power2 msd_2 "0*10*":
```

Next we have to translate (a) into a first-order logic statement. It is just the slightest bit difficult because there is no way to express both i and 2^i in `Walnut`. So instead we make the substitution $b = 2^i$, force b to be a power of 2, and demand that $b \geq 8$ instead of $i \geq 3$. With this substitution, claim (a) can be asserted in `Walnut` as follows:

```
eval evilh "A b,m ($power2(b) & (b>=8) & (m>=1) & (m+3<b)) =>
   (E j,l (j<l) & (l<=b+1) & (m+j=l) & (T[j]=@0) &(T[l]=@0))":
```

which evaluates to `true`. Here `A` is `Walnut`'s way to write the universal quantifier $\forall$.

Similarly, we can translate claim (b). Again we let b stand in for 2^i. Claim (b) then becomes

```
eval odioush "A b,m ($power2(b) & (b>=2) & (m>=1) &(m<b)) =>
   (E j,l (j>=1) & (j<l) & (l<=b) & (m+j=l) & (T[j]=@1)
    & (T[l]=@1))":
```

which also evaluates to `true`. □

We can also combine summands from different sets, provided they are both k-automatic. The *Rudin-Shapiro* numbers

$$3, 6, 11, 12, 13, 15, 19, 22, 24, 25, 26, 30, \ldots$$

are those n for which the number of (possibly overlapping) occurrences of 11 in the base-2 representation of n is odd; they form sequence A022155 in the OEIS. We can describe the natural numbers representable as a sum of an odious number and a Rudin-Shapiro number.

Theorem 5 *Every natural number, except* 0, 1, 2, 3, 6, 9, *is the sum of an odious number and a Rudin-Shapiro number.*

Proof We use the `Walnut` command

```
eval trsum "E i,j (n=i+j) & (T[i]=@1) & (RS[j] = @1)":
```

Here `RS` is a built-in representation of the Rudin-Shapiro sequence. When we enter this command in `Walnut`, it produces the automaton in Fig. 2 accepting the binary representations of n that are sums of the desired form.

A brief inspection shows that this automaton accepts the binary representation of all natural numbers, except 0, 1, 2, 3, 6, 9. □

Finally, we can also apply these ideas to settings that are not strictly covered by Theorem 1. An analogous result holds for sets defined by their Fibonacci representation (and in even more general settings—see [21]). In this case, the automata constructed by `Walnut` are based on the Fibonacci representation of integers, instead of base-k representation.

For example, consider the Wythoff or "Fibonacci-odd" integers

$$1, 4, 6, 9, 12, 14, 17, 19, 22, 25, 27, 30, 33, \ldots$$

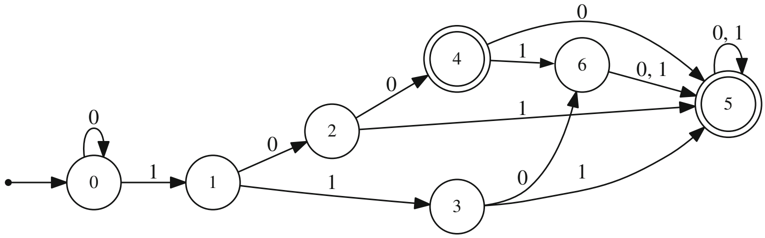

Fig. 2 Automaton for sums of Thue-Morse and Rudin-Shapiro numbers

that form sequence A003622 in the OEIS. These are the analogue of ordinary odd integers (whose base-2 representation ends in 1): they are the integers whose Fibonacci representation ends in 1. We now investigate the additive properties of these numbers.

Theorem 6 *The Fibonacci-odd integers do not form an asymptotic additive basis of order 2. But they do form an asymptotic additive basis of order 3: every number* $N > 23$ *has a representation as a sum of exactly three Fibonacci-odd integers.*

Proof Let us prove the first statement. We construct a `Walnut` statement asserting that n is the sum of exactly two Fibonacci-odd integers:

```
eval sum2f "?msd_fib E i,j (n=i+j) & (F[i]=@1) & (F[j]=@1)":
```

Here the phrase `?msd_fib` indicates that the expression is to be evaluated using the most-significant-digit-first Fibonacci representation, and F is an automaton giving the last digit of the Fibonacci representation (a built-in feature of `Walnut`). The resulting automaton, depicted in Fig. 3, has 12 states and clearly rejects infinitely many integers (for example, those with Fibonacci representation (10)*0. So the Fibonacci-odd integers cannot form an asymptotic additive basis of order 2.

To prove the second statement we write the expression for those numbers that are the sum of three Fibonacci-odd integers:

```
eval sum3f "?msd_fib E i,j,k (n=i+j+k) & (F[i]=@1) & (F[j]=@1)
  & (F[k]=@1)":
```

The resulting automaton, depicted in Fig. 4, has 11 states and, by inspection, accepts the Fibonacci representation of all integers $N > 23$. (Note that the Fibonacci representation of 23 is 1000010.) □

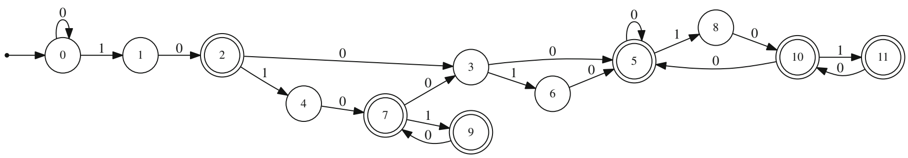

Fig. 3 Automaton for sums of two Fibonacci-odd integers

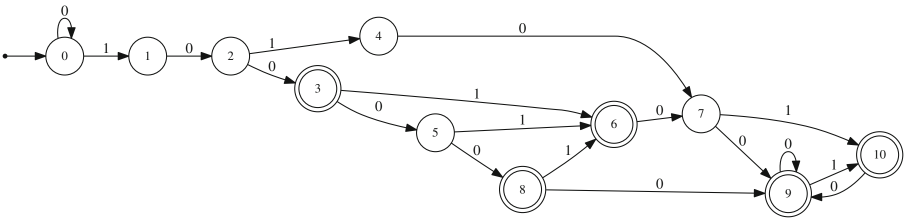

Fig. 4 Automaton for sums of three Fibonacci-odd integers

Our final example involves the difference of two Fibonacci-odd numbers.

Theorem 7 *A natural number n can be represented as the difference of two Fibonacci-odd numbers iff the canonical Fibonacci representation $(n)_F$ is of the form*

$$\epsilon + 10(00 + 10 + 010)^*(\epsilon + 0 + 01).$$

Proof We use the `Walnut` command

```
eval diff2f "?msd_fib E i,j (n+j=i) & (F[i]=@1) & (F[j]=@1)":
```

which gives the automaton depicted in Fig. 5.

We then use the usual state-elimination algorithm on the result, obtaining the regular expression in the statement of the theorem. □

For many of the theorems in this section, one can construct (by hand) alternative, case-based proofs (and some of these proofs are rather simple). Nevertheless, the viewpoint based on automata and first-order logic gives a unified treatment that can handle a wide variety of such claims.

4 The Sum-of-Palindromes Problem

We are interested in integers whose base-k representations are palindromes. In this article, we routinely abuse terminology by calling such an integer a *base-k palindrome*. In the case where $k = 2$, we also call such an integer a *binary palindrome*. The first few binary palindromes are

$$0, 1, 3, 5, 7, 9, 15, 17, 21, 27, 31, 33, 45, 51, 63, \ldots;$$

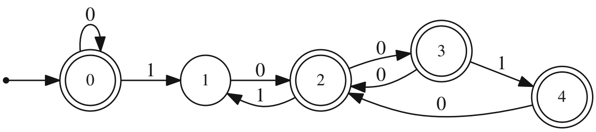

Fig. 5 Automaton for difference of two Fibonacci-odd integers

these form sequence A006995 in the *On-Line Encyclopedia of Integer Sequences* (OEIS).

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

Our goal is to find a constant c such that every natural number is the sum of at most c binary palindromes. To the best of our knowledge, no such bound has been proved up to now. Furthermore, the methods of Section 3 do not apply, since the language of palindromes is not regular. We need a different technique. In Sections 6 and 7 we describe how we can use a decision procedure for nested-word automata to prove the following result:

Theorem 8 *For all $n \geq 8$, every n-bit odd integer is either a binary palindrome itself, or the sum of three binary palindromes*

(a) *of lengths n, $n-2$, and $n-3$; or*
(b) *of lengths $n-1$, $n-2$, and $n-3$.*

As a corollary, we get one of our main results:

Corollary 9 *Every natural number N is the sum of at most 4 binary palindromes.*

Proof It is a routine computation to verify the result for $N < 128$.

Now suppose $N \geq 128$. Let N be an n-bit integer; then $n \geq 8$. If N is odd, then Theorem 8 states that N is the sum of at most 3 binary palindromes. Otherwise, N is even.

If $N = 2^{n-1}$, then it is the sum of $2^{n-1} - 1$ and 1, both of which are palindromes.

Otherwise, $N - 1$ is also an n-bit odd integer. Use Theorem 8 to find a representation for $N - 1$ as the sum of at most 3 binary palindromes, and then add the palindrome 1 to get a representation for N. □

Remark 10 We note that the bound 4 is optimal since, for example, the number 176 is not the sum of two or fewer binary palindromes. (Notice that binary palindromes, other than 0, begin and end with the digit 1, so if an even number is not the sum of two palindromes, at least four palindromes are needed to represent it.)

From our results it follows that the binary palindromes form an additive basis of order four. Recently Julien Cassaigne (personal communication, March 2018) proved that there are infinitely many even natural numbers that are not the sum of two binary palindromes. Thus the binary palindromes do not form an *asymptotic* additive basis of order three.

Sequence A261678 in the OEIS lists those even numbers that are not the sum of two binary palindromes. Sequence A261680 gives the number of distinct representations as the sum of four binary palindromes.

5 Finding an Appropriate Computational Model

To find a suitable model for proving Theorem 8, we turn to formal languages and automata. We seek some class of automata with the following property: for each k, there is an automaton which, given a natural number N as input, accepts the input iff N can be expressed as the sum of k palindromes. Furthermore, we would like the problem of universality ("Does the automaton accept every possible input?") to be decidable in our chosen model. By constructing the appropriate automaton and checking whether it is universal, we could then determine whether every number N can be expressed as the sum of k palindromes.

Palindromes suggest considering the model of pushdown automaton (PDA), since it is well-known that this class of machines, equipped with a stack, can accept the palindrome language $\mathtt{PAL} = \left\{x \in \Sigma^* : x = x^R\right\}$ over any fixed alphabet Σ. A tentative approach is as follows: create a PDA M that, on input N expressed in base 2, uses nondeterminism to "guess" the k summands and verify that (a) every summand is a palindrome, and (b) they sum to the input N. We would then check to see if M accepts all of its inputs. However, two problems immediately arise.

The first problem is that universality is recursively unsolvable for nondeterministic PDAs [45, Thm. 8.11, p. 203], so even if the automaton M existed, there would be no algorithm guaranteed to check its universality.

The second problem involves checking that the guessed summands are palindromes. One can imagine guessing the summands in parallel, or in series. If we try to check them in parallel, this seems to correspond to the recognition of a language which is not a CFL (i.e., a context-free language, the class of languages recognized by nondeterministic PDAs). Specifically, we encounter the following obstacle:

Theorem 11 *The set L of words over the alphabet $\Sigma \times (\Sigma \cup \#)$, where the first "track" is a palindrome and the second "track" is a another, possibly shorter, palindrome, padded on the right with # signs, is not a CFL.*

Proof Assume that it is. Consider L intersected with the regular language

$$[1,1]^+[1,0][1,1]^+[0,1][1,\#]^+,$$

and call the result L'. We use Ogden's lemma [62] to show L' is not a CFL.

Choose z to be a word where the first track is $\left(1^{2n}01^{2n}\right)$ and the second track is $(1^n01^n\#^{2n})$. Mark the compound symbols $[1,\#]$. Then every factorization $z = uvwxy$ must have at least one $[1,\#]$ in v or x. If it is in v, then the only choice for x is also $[1,\#]$, so pumping gives a non-palindrome on the first track. If it is in x, then v can be $[1,1]^i$, or contain $[1,0]$ or $[0,1]$. If the latter, pumping twice gives a word not in L' because there is more than one 0 on one of the two tracks. If the former, pumping twice gives a word with the second track not a palindrome. This contradiction shows that L', and hence L, is not a context-free language. □

So, using a pushdown automaton, we cannot check two arbitrary palindromes of wildly unequal lengths in parallel.

If the summands were presented serially, we could check whether each summand individually is a palindrome, using the stack, but doing so destroys our copy of the summand, and so we cannot add them all up and compare them to the input. In fact, we cannot add serial summands in any case, because we have

Theorem 12 *The language*

$$L = \{(m)_2\#(n)_2\#(m+n)_2 \ : \ m, n \geq 0\}$$

is not a CFL.

Proof Assume L is a CFL and intersect with the regular language $1^+01^+\#1^+\#1^+0$, obtaining L'. We claim that

$$L' = \{1^a01^b\#1^c\#1^d0 \ : \ b = c \text{ and } a + b = d\}.$$

This amounts to the claim, easily verified, that the only solutions to the equation $2^{a+b+1} - 2^b - 1 + 2^c - 1 = 2^{d+1} - 2$ are $b = c$ and $a + b = d$. Then, starting with the word $z = 1^n01^n\#1^n\#1^{2n}0$, an easy argument with Ogden's lemma proves that L' is not a CFL, and hence neither is L. □

So, using a pushdown automaton, we cannot handle summands given in series, either.

These issues lead us to restrict our attention to representations as sums of palindromes of the same (or similar) lengths. More precisely, we consider the following variant of the additive problem of palindromes: for a given number of summands k, a a bound b, and a natural number N as input, is N the sum of k palindromes whose lengths differ by at most b? Since the palindromes are all of similar lengths, a stack can be used to guess and verify the palindromes in parallel. To tackle this problem, we need a model which is both (1) powerful enough to handle our new variant, and (2) restricted enough that universality is decidable. We find such a model in the class of *nested-word automata*, described in the next section.

6 Restating the Problem in the Language of Nested-word Automata

Nested-word automata (NWAs) were popularized by Alur and Madhusudan [3, 4], although essentially the same model was discussed previously by Mehlhorn [58], von Braunmühl and Verbeek [20], and Dymond [32]. They are a restricted variant of pushdown automata. Readers familiar with visibly pushdown automata (VPA) should note that NWAs are equivalent in power to VPAs [3, 4]. We only briefly describe their functionality here. For other theoretical aspects of nested-word and visibly-pushdown automata, see [38, 51, 63, 64, 68].

The input alphabet of an NWA is partitioned into three sets: a *call alphabet*, an *internal alphabet*, and a *return alphabet*. An NWA has a stack, but has more restricted access to it than PDAs do. If an input symbol is from the internal alphabet, the NWA cannot access the stack in any way. If the input symbol read is from the call alphabet, the NWA pushes its current state onto the stack, and then performs a transition, based only on the current state and input symbol read. If the input symbol read is from the return alphabet, the NWA pops the state at the top of the stack, and then performs a transition based on three pieces of information: the current state, the popped state, and the input state read. An NWA accepts if the state it terminates in is an accepting state.

As an example, Fig. 6 illustrates a nested-word automaton accepting the language $\{0^n 1 2^n \ : \ n \geq 1\}$.

Here the call alphabet is $\{0\}$, the internal alphabet is $\{1\}$, and the return alphabet is $\{2\}$.

Nondeterministic NWAs are a good machine model for our problem, because nondeterminism allows "guessing" the palindromes that might sum to the input, and the stack allows us to "verify" that they are indeed palindromes. Deterministic NWAs are as expressive as nondeterministic NWAs, and the class of languages they accept is closed under the operations of union, complement and intersection. Finally, testing emptiness, universality, and language inclusion are all decidable problems for NWAs [3, 4]; there is an algorithm for each of them.

For a nondeterministic NWA of n states, the corresponding determinized machine has at most $2^{\Theta(n^2)}$ states, and there are examples for which this bound is attained. This very rapid explosion in state complexity potentially could make deciding problems such as language inclusion infeasible in practice. Fortunately, we did not run into determinized machines with more than 40000 states in proving our results. Most of the algorithms invoked to prove our results run in under a minute.

We now discuss the general construction of the NWAs that check whether inputs are sums of binary palindromes. We partition the input alphabet into the call alphabet $\{a, b\}$, the internal alphabet $\{c, d\}$, and the return alphabet $\{e, f\}$. The symbols a, c, and e correspond to 0, while b, d, and f correspond to 1. The input word is fed to the machine starting with the *least significant digit*. We provide the NWA with input words whose first half is entirely made of call symbols, and second half is entirely made of return symbols. Internal symbols are used to create a divider between the halves (for the case of odd-length inputs).

The idea behind the NWA is to nondeterministically guess all possible summands when reading the first half of the input word. The guessed summands are characterized by the states pushed onto the stack. The machine then checks if the guessed summands can produce the input bits in the second half of the word. The machine keeps track of any carries in the current state.

The general procedure to prove our results is to build an NWA `PalSum` accepting only those inputs that it verifies as being appropriate sums of palindromes, as well

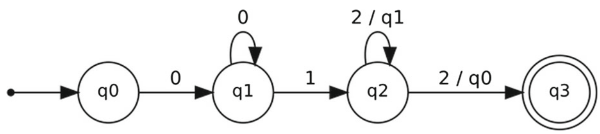

Fig. 6 A nested-word automaton for the language $\{0^n 1 2^n \ : \ n \geq 1\}$

as an NWA `SyntaxChecker` accepting all valid representations. We then run the decision algorithms for language inclusion, language emptiness, etc. on `PalSum` and `SyntaxChecker` as needed. To do this, we used the Automata Library toolchain of the ULTIMATE program analysis framework [42, 43].

We have provided links to the proof scripts used to establish all of our results. To run these proof scripts, simply copy the contents of the script into https://monteverdi.informatik.uni-freiburg.de/tomcat/Website/?ui=int&tool=automata_library and click "execute".

7 Proving Theorem 8

In this section, we discuss construction of the appropriate nested-word automaton in more detail.

Proof (of Theorem 8)

We build three separate automata. The first, `palChecker`, has 9 states, and simply checks whether the input number is a binary palindrome. The second, `palChecker2`, has 771 states, and checks whether an input number of length n can be expressed as the sum of three binary palindromes of lengths n, $n-2$, and $n-3$. The third machine, `palChecker3`, has 1539 states, and checks whether an input number of length n can be expressed as the sum of three binary palindromes of lengths $n-1$, $n-2$, and $n-3$. We then determinize these three machines, and take their union, to get a single deterministic NWA, `FinalAut`, with 36194 states.

The language of valid inputs to our automata is given by

$$L = \{\{a, b\}^n \{c, d\}^m \{e, f\}^n \ : \ 0 \leq m \leq 1, \ n \geq 4\}.$$

We only detail the mechanism of `palChecker3` here. Let p, q and r be the binary palindromes representing the guessed n-length summand, $(n-2)$-length summand and $(n-3)$-length summand respectively. The states of `palChecker3` include 1536 t-states that are 10-tuples. We label these states $(g, x, y, z, k, l_1, l_2, m_1, m_2, m_3)$, where $g \in \{0, 1, 2\}$, while all other coordinates are either 0 or 1. The g-coordinate indicates the current carry, and can be as large as 2. The x, y and z coordinates indicate whether we are going to guess 0 or 1 for the next guesses of p, q and r respectively. The remaining coordinates serve as "memory" to help us manage the differences in lengths of the guessed summands. The k-coordinate remembers the most recent guess for p. We have l_1 and l_2 remember the two most recent q guesses, with l_1 being the most recent one, and we have m_1, m_2 and m_3 remember the three most recent r-guesses, with m_1 being the most recent one, then m_2, and m_3 being the guess we made three steps ago. We also have three s-states labeled s_0, s_1 and s_2, representing carries of 0, 1 and 2 respectively. These states process the second half of the input word.

The initial state of the machine is $(0, 1, 1, 1, 0, 0, 0, 0, 0, 0)$ since we start with no carry, must guess 1 for our first guess of a valid binary palindrome, and all "previous" guesses are 0. A t-state has an outgoing transition on either a or b, but not both. If $g + x + y + z$ produces an output bit of 0, it takes a transition on a, else it takes a transition on b. The destination states are all six states of the form $(g', x', y', z', x, y, l_1, z, m_1, m_2)$, where g' is the carry resulting from $g+x+y+z$, and x', y', z' can be either 0 or 1. Note that we "update" the remembered states by forgetting k, l_2 and m_3, and saving x, y and z.

The s-states only have transitions on the return symbols e and f. When we read these symbols, we pop a t-state off the stack. If state s_i pops the state $(g, x, y, z, k, l_1, l_2, m_1, m_2, m_3)$ off the stack, its transition depends on the addition of $i + k + l_2 + m_3$. If this addition produces a carry of j, then s_i can take a transition to s_j on e if the output bit produced is 0, and on f otherwise. By reading the k, l_2 and m_3 values, we correctly realign p, q and r, correcting for their different lengths. This also ensures that we supply 0s for the last three guesses of r, the last two guesses of q and the last guess of p. The only accepting state is s_0.

It remains to describe how we transition from t-states to s-states. This transition happens when we are halfway done reading the input. If the length of the input is odd, we read a c or a d at the halfway point. We only allow certain t-states to have outgoing transitions on c and d. Specifically, we require the state's coordinates to satisfy $k = x$, $l_2 = y$, $m_1 = m_2$ and $m_3 = z$. These conditions are required for p, q and r to be palindromes. We transition to state s_i if the carry produced by adding $g+x+y+z$ is i, and we label the transition c if the output bit is 0, and d otherwise.

If the length of the input is even, then our t-states take a return transition on e or f. Once again, we restrict the t-states that can take return transitions. We require the state's coordinates to satisfy $l_1 = l_2$ and $m_1 = m_3$ to ensure our guessed summands are palindromes. Let the current state be $(g, x, y, z, k, l_1, l_1, m_1, m_2, m_1)$, and the state at the top of the stack be $\left(g', x', y', z', k', l_1', l_2', m_1', m_2', m_3'\right)$. We can take a transition to s_i if the sum $g + k' + l_2' + m_3'$ produces a carry of i, and we label the transition e if the output bit is 0, and f otherwise.

The structure and behavior of `palChecker2` is very similar. One difference is that there is no need for a k-coordinate in the t-states since the longest summand guessed is of the same length as the input.

The complete script executing this proof is over 750000 lines long. Since these automata are very large, we wrote two C++ programs to generate them. Both the proof script, and the programs generating them can be found at https://github.com/arajasek/FinalGenerators. It is worth noting that a t-state labeled as $(g, x, y, z, k, l_1, l_2, m_1, m_2, m_3)$ in this paper is labeled $q\_g\_xyz\_k\_l_1l_2\_m_1m_2m_3$ in the proof script. Also, ULTIMATE does not currently have a union operation for NWAs, so we work around this by using De Morgan's laws for complement and intersection. □

To check our implementation, we built an NWA-simulator, and then ran simulations of the machines on various types of inputs, which we then checked against experimental results. For instance, we built the machine that accepts representations of integers that can be expressed as the sum of 2 binary palindromes. We then simulated this machine on every integer from 513 to 1024, and checked that it only accepts those integers that we experimentally confirmed as being the sums of 2 binary palindromes.

8 Bases 3 and 4

In this section we prove analogous results for bases 3 and 4. Our main results are as follows:

Theorem 13 *Every natural number is the sum of at most three base-3 palindromes.*

Theorem 14 *Every natural number is the sum of at most three base-4 palindromes.*

We tried to prove these theorems using the method, based on NWAs, described in previous sections. However, the NWAs needed became too large for us to manipulate effectively. Instead, we use a modified approach using nondeterministic finite automata to prove these results. This approach was suggested to us by Dirk Nowotka and Parthasarathy Madhusudan, independently.

We start with Theorem 13. Our proof is based on the following lemma.

Lemma 15 *For all $n \geq 9$, every integer whose base-3 representation is of length n is the sum of*

(a) *three base-3 palindromes of lengths n, $n-1$, and $n-2$; or*
(b) *three base-3 palindromes of lengths n, $n-2$, and $n-3$; or*
(c) *three base-3 palindromes of lengths $n-1$, $n-2$, and $n-3$; or*
(d) *two base-3 palindromes of lengths $n-1$ and $n-2$.*

Proof We represent the input in a "folded" manner over the input alphabet $\Sigma_3 \cup (\Sigma_3 \times \Sigma_3)$, where $\Sigma_k = \{0, 1, \ldots, k-1\}$, giving the machine 2 letters at a time from opposite ends. This way we can directly guess our summands without having need of a stack at all. We align the input along the $n-2$ summand by providing the first 2 letters of the input separately.

If $(N)_3 = a_{2i+1}a_{2i} \cdots a_0$, we represent N as the word

$$a_{2i+1}a_{2i}[a_{2i-1}, a_0][a_{2i-2}, a_1] \cdots [a_i, a_{i-1}].$$

As an example, $(5274)_3 = 21020100$ is represented as 2, 1, [0, 0], [2, 0], [0, 1].

Odd-length inputs leave a trailing unfolded letter at the end of their input. If $(N)_3 = a_{2i+2}a_{2i+1} \cdots a_0$, we represent N as the word

$$a_{2i+2}a_{2i+1}[a_{2i}, a_0][a_{2i-1}, a_1] \cdots [a_{i+1}, a_{i-1}]a_i.$$

As an example, $(15823)_3 = 210201001$ is represented as 2, 1, [0, 1], [2, 0], [0, 0], 1.

We need to simultaneously carry out addition on both ends. In order to do this we need to keep track of two carries. On the lower end, we track the "incoming" carry at the start of an addition, as we did in the proofs using NWAs. On the higher end, however, we track the expected "outgoing" carry.

To illustrate how our machines work, we consider an NFA accepting length-n inputs that are the sum of 4 base-3 palindromes, one each of lengths n, $n-1$, $n-2$ and $n-3$. Although this is not a case in our theorem, each of the four cases in our theorem can be obtained from this machine by striking out one or more of the guessed summands.

Recall that we aligned our input along the length-(n–2) summand by providing the 2 most significant letters in an unfolded manner. This means that our guess for the length-n summand will be "off-by-two": when we make a guess at the higher end of the length-n palindromic summand, its appearance at the lower end is 2 steps away. We hence need to remember the last 2 guesses at the higher end of the length-n summand in our state. Similarly, we need to remember the most recent higher guess of the length-$(n-1)$ summand, since it is off-by-one. The length-$(n-2)$ summand is perfectly aligned, and hence nothing needs to be remembered. The length-$(n-3)$ summand has the opposite problem of the length-$(n-1)$ input. Its lower guess only appears at the higher end one step later, and so we save the most recent guess at the lower end.

Thus, in this machine, we keep track of 6 pieces of information:

- c_1, the carry we are expected to produce on the higher end,
- c_2, the carry we have entering the lower end,
- x_1 and x_2, the most recent higher guesses of the length-n summand,
- y, the most recent higher guess of the length-$(n-1)$ summand, and
- z, the most recent lower guess of the length-$(n-3)$ summand,

Consider a state $(c_1, c_2, x_1, x_2, y, z)$. Let $0 \leq i, j, k, l < 3$ be our next guesses for the four summands of lengths $n, n-1, n-2$ and $n-3$ respectively. Also, let α be our guess for the next incoming carry on the higher end. Let the result of adding $i + j + k + z + \alpha$ be a value $0 \leq p_1 < 3$ and a carry of q_1. Let the result of adding $x + y + k + l + x_2$ be a value $0 \leq p_2 < 3$ and a carry of q_2. We must have $q_1 = c_1$. If this condition is met, we add a transition from this state to $(\alpha, q_2, x_2, i, j, l)$, and label the transition $[p_1, p_2]$.

The initial state is (0, 0, 0, 0, 0, 0). We expand the alphabet to include special variables for the first 3 symbols of the input word. This is to ensure that we always guess a 1 or a 2 for the first (and last) positions of our summands.

The acceptance conditions depend on whether $(N)_3$ is of even or odd length. If a state $(c_1, c_2, x_1, x_2, y, z)$ satisfies $c_1 = c_2$ and $x_1 = x_2$, we set it as an accepting states. A run can only terminate in one of these states if $(N)_3$ is of even length. We accept since we are confident that our guessed summands are palindromes (the condition $x_1 = x_2$ ensures our length-n summand is palindromic), and since the last outgoing carry on the lower end is the expected first incoming carry on the higher end (enforced by $c_1 = c_2$).

We also have a special symbol to indicate the trailing symbol of an input for which $(N)_3$ is of odd length. We add transitions from our states to a special accepting state, q_{acc}, if we read this special symbol. Consider a state $(c_1, c_2, x_1, x_2, y, z)$, and let $0 \leq k < 3$ be our middle guess for the $n-2$ summand. Let the result of adding $x_1 + y + k + z + c_2$ be a value $0 \leq p < 3$ and a carry of q. If $q = c_1$, we add a transition on p from our state to q_{acc}.

We wrote a C++ program generating the machines for each case of this theorem. After taking the union of these machines, determinizing and minimizing, the machine has 378 states. We then built a second NFA that accepts folded representations of $(N)_3$ such that the unfolded length of $(N)_3$ is greater than 8. We then use ULTIMATE to assert that the language accepted by the second NFA is included in that accepted by the first. All these operations run in under a minute.

We tested this machine by experimentally calculating which values of $243 \leq N \leq 1000$ could be written as the sum of palindromes satisfying one of our 4 conditions. We then asserted that for all the folded representations of $243 \leq N \leq 1000$, our machine accepts these values which we experimentally calculated, and rejects all others. □

Now that Lemma 15 is proved, Theorem 13 follow almost immediately.

Proof of Theorem 13 For $N < 6561$ we can easily verify by an explicit computation that the result is true. Otherwise, $(N)_3$ is of length at least 9, and we can apply Corollary 15. □

For base 4 we have the following:

Lemma 16 *For all $n \geq 7$, every integer whose base-4 representation is of length n is the sum of*

(a) exactly one palindrome each of lengths $n-1$, $n-2$, and $n-3$; or
(b) exactly one palindrome each of lengths n, $n-2$, and $n-3$.

Proof The NFA we build is very similar to the machine described for the base-3 proof. Indeed, the generator used is the same as the one for the base-3 proof, except that its input base is 4, and the only machines it generates are for the two cases of this theorem. The minimized machine has 478 states. □

Proof Proof of Theorem 14 If $N < 4096$ then we can verify the result directly. Otherwise $(N)_4$ has length 7 or more, and we can apply Lemma 16. □

Theorems 13 and 14, together with the results previously obtained by Cilleruelo, Luca, & Baxter, complete the additive theory of palindromes for all integer bases $b \geq 2$.

Of course, this approach using NFAs would also work for base 2.

An advantage to this approach is the following result:

Theorem 17 *Given an integer $N \geq 0$ we can, in time linear in $\log N$, determine a representation for N as the sum of four binary palindromes (resp., three base-3 palindromes, three base-4 palindromes).*

Proof Once the NFAs have been constructed as above, we can carry out the usual direct product construction with a DFA of $O(\log N)$ states accepting the single word representing N in base 2 (resp., bases 3 and 4). We then use depth-first search to find a path to an accepting state of the resulting automaton. The labels of this path give the desired representation. □

9 Variations on the Palindrome Problem

In this section we consider some variations on the palindrome problem. Our first variation involves the notion of generalized palindromes.

9.1 Generalized Palindromes

We define a *generalized palindrome of length n* to be an integer whose base-k representation, extended, if necessary, by adding leading zeroes to make it of length n, is a palindrome. For example, 12 is a generalized binary palindrome of length 6, since its length-6 representation 001100 is a palindrome. If a number is a generalized palindrome of any length, we call it a *generalized palindrome*. The first few binary generalized palindromes are

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, 16, 17, 18, 20, 21, 24, 27, 28, 30, 31, 32, \ldots;$$

they form sequence A057890 in the OEIS.

We can use our method to prove

Theorem 18 *Every natural number is the sum of at most 3 generalized binary palindromes.*

Proof For $N < 32$, the result can easily be checked by hand.

We then use our method to prove the following claim:

Every length-n integer, $n \geq 6$, is the sum of at most two generalized palindromes of length n and one of length $n-1$.

We build a single NWA, `gpalChecker`, accepting representations of length-n integers that can be written as the sum of at most two generalized palindromes of length n and one of length $n-1$. The structure of `gpalChecker` is a highly simplified version of the one described in Section 7. With generalized palindromes we allow the first guessed bit to be zero. We do not have to worry about the "at most" clause in our claim, since the word consisting of n 0s is a valid generalized palindrome. The nondeterministic machine has 39 states, which grows to 832 states on determinization. As before, we build an NWA `syntaxChecker` that accepts all valid inputs, and assert that the language accepted by `syntaxChecker` is included in that accepted by `gpalChecker`.

The complete proof script, as well as the C++ program generating it, can be found at https://github.com/arajasek/FinalGenerators. □

As an aside, we also mention the following enumeration result.

Theorem 19 *There are exactly $3^{\lceil n/2 \rceil}$ natural numbers that are the sum of two generalized binary palindromes of the same length n.*

Proof Take two generalized binary palindromes of length n and add them bit-by-bit. Then each digit position is either 0, 1, or 2, and the result is still a palindrome. Hence there are at most $3^{\lceil n/2 \rceil}$ such numbers. It remains to see they are all distinct.

We show that if x is a length-n word that is a palindrome over $\{0, 1, 2\}$, then the map that sends x to its evaluation in base 2 is injective.

We do this by induction on the length of the palindrome. The claim is easily verified for length 0 and 1. Now suppose x and y are two distinct words of the same length that evaluate to the same number in base 2. Then, by considering everything mod 2, we see that either

(a) x and y both end in the same number i, or
(b) (say) x ends in 0 and y ends in 2.

In case (a) we know that x and y, since they are both palindromes, both begin and end in i. So subtracting $i00\cdots0i$ from x and y we get words x', y' of length $n-2$ that evaluate to the same thing, and we can apply induction to see $x' = y'$.

In case (b), say $x = 0x'0$ and $y = 2y'2$, and they evaluate to the same thing. However, the largest x could be is $0222\cdots20$ and the smallest y could be is $200\cdots02$. The difference between these two numbers is 6, so we conclude that this is impossible. □

Remark 20 Not every natural number is the sum of 2 generalized binary palindromes; the smallest exception is 157441. The list of exceptions forms sequence A278862 in the OEIS. It does not seem to be known currently whether there are infinitely many exceptions.

9.2 Antipalindromes

Another generalization concerns antipalindromes. Define a map from Σ_2^* to Σ_2^* by changing every 0 to 1 and vice-versa; this is denoted $\overline{w}$. Thus, for example, $\overline{010} = 101$. An *antipalindrome* is a word w such that $w = \overline{w^R}$. It is easy to see that if a word w is an antipalindrome, then it must be of even length. We call an integer n an antipalindrome if its canonical base-2 expansion is an antipalindrome. The first few antipalindromic integers are

$$2, 10, 12, 38, 42, 52, 56, 142, 150, 170, 178, 204, 212, 232, 240, \ldots;$$

they form sequence A035928 in the OEIS. Evidently every antipalindrome must be even.

To use NWAs to prove results about antipalindromes, we slightly modify our machine structure. While processing the second portion of the input word, we complement the number of ones we guessed in the first portion. Assume we are working with n summands, and let the input symbol be $q \in \{e, f\}$, and the state at the top of the stack be $f_{x,y}$. The state s_i has a transition to s_j if $i + (n - y)$ produces an output bit corresponding to q and a new carry of j. We also set the initial state to be $f_{0,0}$, because the least significant bit of an antipalindrome must be 0.

Using our method, we would like to prove the following conjecture. However, so far we have been unable to complete the computations because our program runs out of space. In a future version of this paper, we hope to change the status of the following conjecture to a theorem.

Conjecture 21 Every even integer of length n, for n odd and $n \geq 9$, is the sum of at most 10 antipalindromes of length $n - 3$.

Corollary 22 *(Conditional only on the truth of Conjecture 21) Every even natural number N is the sum of at most 13 antipalindromes.*

Proof For $N < 256$ this can be verified by a short program. (In fact, for these n, only 4 antipalindromes are needed.) Otherwise assume $N \geq 256$. Then N is of length $n \geq 9$. If n is odd, we are done by Theorem 21. Otherwise, n is even. Then, by subtracting at most 3 copies of the antipalindrome $2^{n-2} - 2^{\frac{n}{2}-1}$ from N, we obtain N' even, of length $n-1$. We can then apply Theorem 21 to N'. □

The 13 in Corollary 22 is probably quite far from the optimal bound. Numerical evidence suggests

Conjecture 23 Every even natural number is the sum of at most 4 antipalindromes.

Conjecture 24 Every even natural number except

8, 18, 28, 130, 134, 138, 148, 158, 176, 318, 530, 538, 548, 576, 644, 1300, 2170, 2202, 2212, 2228, 2230, 2248, 8706, 8938, 8948, 34970, 35082

is the sum of at most 3 antipalindromes.

9.3 Generalized Antipalindromes

Similarly, one could consider "generalized antipalindromes"; these are numbers whose base-2 expansion becomes an antipalindrome if a suitable number of leading zeroes are added. The notion of length here is the same as in Section 9.1.

Theorem 25 *Every natural number of length n, for $n \geq 6$ and even, is the sum of exactly 6 generalized antipalindromes of length $n-2$.*

Proof Since generalized antipalindromes can have leading zeroes, we allow all f-states with no carry as initial states. We also complement the number of ones for the second half, as we did when handling regular antipalindromes.

The automated proof of (a) can be found at https://github.com/arajasek/FinalGenerators. The determinized automaton has 2254 states. □

Corollary 26 *Every natural number is the sum of at most 7 generalized antipalindromes.*

Proof Just like the proof of Corollary 9, using Theorem 25. □

Remark 27 Corollary 26 is probably not best possible. The correct bound seems to be 3. The first few numbers that do not have a representation as the sum of 2 generalized antipalindromes are

29, 60, 91, 109, 111, 121, 122, 131, 135, 272, 329, 347, 365, 371, 373, 391, 401, 429, 441, 445, 449, 469, 473, 509, 531, 539, 546, 577, 611, 660, 696, 731, 744, 791, 804, 884, 905, 940, 985, 1011, 1020, 1045, ...

9.4 Fibonacci Palindromes

Other variations include using other kinds of representations. For example, in Fibonacci representation [52, 79], integers are represented as a sum of Fibonacci numbers

$$\sum_{0 \le i \le t} e_i F_{i+2},$$

(where $e_i \in \{0, 1\}$ and $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$. To achieve uniqueness, we impose the additional constraint that $e_i e_{i+1} = 0$. Fibonacci palindromes are those whose Fibonacci representation (with no leading zeroes) is a palindrome; they form sequence A094202 in the OEIS.

In principle, our methods could probably be used to prove the following.

Conjecture 28 Every natural number except 2429, 7699, and 8597 is the sum of at most three Fibonacci palindromes. For these numbers four summands are required.

We have verified this conjecture up to 10^7.

10 Objections to These Kinds of Proofs

A proof based on computer calculations, like the one we have presented here, is occasionally criticized because it cannot easily be verified by hand, and because it relies on software that has not been formally proved. These kinds of criticisms are not new; they date at least to the 1970's, in response to the celebrated proof of the four-color theorem by Appel and Haken [5, 6]. See, for example, Tymoczko [76].

We answer this criticism in several ways. First, it is not reasonable to expect that every result of interest to mathematicians will have short and simple proofs. There may well be, for example, easily-stated results for which the shortest proof possible in a given axiom system is longer than any human mathematician could verify in their lifetime, even if every waking hour were devoted to checking it. For these kinds of results, an automated checker may be our only hope. There are many results for which the only proof currently known is computational.

Second, while short proofs can easily be checked by hand, what guarantee is there that any very long case-based proof — whether constructed by humans or computers — can always be certified by human checkers with a high degree of confidence? There is always the potential that some case has been overlooked. Indeed, the original proof by Appel and Haken apparently overlooked some cases. Similarly, the original proof by Cilleruelo & Luca on sums of palindromes [25] had some minor flaws that became apparent once their method was implemented as a `python` program.

Third, confidence in the correctness of the results can be improved by providing code that others may check. Transparency is essential. To this end, we have provided our code for the automata we discussed, and the reader can easily run this code on the software we referenced. Our code is available at https://github.com/arajasek/FinalGenerators.

Another objection is that these kinds of proofs necessarily depend on a particular fixed base. There is no evident way to extend our technique to apply to infinitely many bases at once.

11 Schnirelmann Density

Another, more number-theoretic, approach to the problems we discuss in this paper is Schnirelmann density. Given a set $S \subseteq \mathbb{N}$, define $A_S(x) = \sum_{\substack{i \in S \\ 1 \le i \le x}} 1$ to be the counting function associated with S. The Schnirelmann density of S is then defined to be

$$\sigma(S) := \inf_{n \geq 1} \frac{A_s(n)}{n}.$$

Classical theorems of additive number theory (e.g., [61, §7.4]) relate the property of being an additive basis to the value of $\sigma(S)$. We pose the following open problems:

Open Problem 29 What is the Schnirelmann density d_k of those numbers expressible as the sum of at most k binary palindromes? By computation we find $d_2 < 0.443503$ and $d_3 < .942523$.

Open Problem 30 Let A be a k-automatic set of natural numbers, or its analogue using a pushdown automaton or nested-word automaton. Is $\sigma(A)$ computable?

12 Moral of the Story

We conclude with the following thesis, expressed as two principles.

1. If an argument is heavily case-based, consider turning the proof into an algorithm.
2. If an argument is heavily case-based, seek a logical system or machine model where the assertions can be expressed, and prove them purely mechanically using a decision procedure.

Can other new results in number theory or combinatorics be proved using our approach? We have proved results about binary squares in [56]. Obtaining other results is left as a challenge for the reader.

Acknowledgments We thank Dirk Nowotka, Parthasarathy Madhusudan, and Jean-Paul Allouche for helpful discussions. We thank the creators of the ULTIMATE automaton library for their assistance. We are also grateful to the referees for reading our paper with care and making several helpful suggestions.

References

1. Allouche, J.-P., Rampersad, N., Shallit, J.: Periodicity, repetitions, and orbits of an automatic sequence. Theoret. Comput. Sci. **410**, 2795–2803 (2009)

2. Aloui, K., Mauduit, C., Mkaouar, M.: Somme des chiffres et répartition dans les classes de congruence pour les palindromes ellipséphiques. Acta Math. Hung. **151**, 409–455 (2017)
3. Alur, R., Madhusudan, P.: Visibly Pushdown Languages. In: Proc. Thirty-Sixth Ann. ACM Symp. Theor. Comput. (STOC), pp. 202–211, ACM (2004)
4. Alur, R., Madhusudan, P.: Adding nested structure to words. J. Assoc. Comput. Mach. 56 (2009), Art. 16
5. Appel, K., Haken, W.: Every planar map is four colorable. I. Discharging Illinois J. Math. **21**, 429–490 (1977)
6. Appel, K., Haken, W., Koch, J.: Every planar map is four colorable. II. Reducibility Illinois J. Math. **21**, 491–567 (1977)
7. Ashbacher, C.: More on palindromic squares. J. Recreat. Math. **22**, 133–135 (1990)
8. Banks, W.D.: Every natural number is the sum of forty-nine palindromes. INTEGERS — Electronic J. Combinat. Number Theory 16 (2016), #A3 (electronic)
9. Banks, W.D., Hart, D.N., Sakata, M.: Almost all palindromes are composite. Math. Res. Lett. **11**, 853–868 (2004)
10. Banks, W.D., Shparlinski, I.E.: Prime divisors of palindromes. Period. Math Hung. **51**, 1–10 (2005)
11. Banks, W.D., Shparlinski, I.E.: Average value of the Euler function on binary palindromes. Bull. Pol. Acad. Sci Math. **54**, 95–101 (2006)
12. Barnett, J.K.R.: Tables of square palindromes in bases 2 and 10. J. Recreat. Math. **23**(1), 13–18 (1991)
13. Basić, B.: On d-digit palindromes in different bases: the number of bases is unbounded. Internat. J. Number Theory **8**, 1387–1390 (2012)
14. Basić, B.: On "very palindromic" sequences. J. Korean Math. Soc. **52**, 765–780 (2015)
15. Bell, J., Hare, K., Shallit, J.: When is an automatic set an additive basis. Proc. Amer. Math. Soc. Ser. B **5**, 50–63 (2018)
16. Bell, J.P., Lidbetter, T.F., Shallit, J.: Additive number theory via approximation by regular languages. In: Hoshi, M., Seki, S. (eds.) DLT Vol. 11088 of Lecture Notes in Computer Science, pp. 121–132. Springer, Berlin (2018)
17. Bérczes, A., Ziegler, V.: On simultaneous palindromes. J. Combin. Number Theory **6**, 37–49 (2014)
18. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning Ways for your Mathematical Plays, Vol. 2 Games in Particular. Academic Press, Cambridge (1982)
19. Berstel, J.: Axel Thue's Papers on Repetitions in Words: a Translation. Number 20 in Publications Du Laboratoire De Combinatoire Et d'Informatique Mathématique. Université du Québec à Montréal (1995)
20. von Braunmühl, B., Verbeek, R.: Input-driven languages are recognized in n space. In: Karpinski, M. (ed.) Foundations of Computation Theory, FCT 83, Vol. 158 of Lecture Notes in Computer Science, pp. 40–51. Springer, Berlin (1983)
21. Bruyère, V., Hansel, G.: Bertrand numeration systems and recognizability. Theoret. Comput. Sci. **181**, 17–43 (1997)
22. Bruyère, V., Hansel, G., Michaux, C., Villemaire, R.: Logic and p-recognizable sets of integers. Bull. Belgian Math. Soc. **1**, 191–238 (1994). Corrigendum, Bull. Belg. Math. Soc. 1 (1994) 577
23. Büchi, J.R.: Weak secord-order arithmetic and finite automata. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **6**, 66–92 (1960). Reprinted in S. Mac Lane and D. Siefkes, eds., *The Collected Works of J. Richard Büchi*, Springer, 1990, pp.398–424
24. Charlier, E., Rampersad, N., Shallit, J.: Enumeration and decidable properties of automatic sequences. Internat. J. Found. Comp. Sci. **23**, 1035–1066 (2012)
25. Cilleruelo, J., Luca, F.: Every positive integer is a sum of three palindromes. arXiv:1602.06208v1 (2016)
26. Cilleruelo, J., Luca, F., Baxter, L.: Every positive integer is a sum of three palindromes. Math. Comput. **87**, 3023–3055 (2018)
27. Cilleruelo, J., Luca, F., Shparlinski, I.E.: Power values of palindromes. J Combin. Number Theory **1**, 101–107 (2009)
28. Cilleruelo, J., Tesoro, R., Luca, F.: Palindromes in linear recurrence sequences. Monatsh. Math. **171**, 433–442 (2013)
29. Col, S.: Palindromes dans les progressions arithmétiques. Acta Arith. **137**, 1–41 (2009)
30. Du, C.F., Mousavi, H., Rowland, E., Schaeffer, L.: J. Shallit. Decision algorithms for Fibonacci-automatic words II: related sequences and avoidability. Theoret. Comput. Sci. **657**, 146–162 (2017)

31. Du, C.F., Mousavi, H., Schaeffer, L., Shallit, J.: Decision algorithms for Fibonacci-automatic words III: enumeration and abelian properties. Internat. J. Found. Comp. Sci. **27**, 943–963 (2016)
32. Dymond, P.W.: Input-driven languages are in n depth. Inform. Process. Lett. **26**, 247–250 (1988)
33. Goč, D., Henshall, D., Shallit, J.: Automatic theorem-proving in combinatorics on words. In: Moreira, N., Reis, R. (eds.) CIAA 2012, Vol. 7381 of Lecture Notes in Computer Science, pp. 180–191. Springer, Berlin (2012)
34. Goč, D., Mousavi, H., Shallit, J.: On the number of unbordered factors. In: Dediu, A.-H., Martin-Vide, C., Truthe, B. (eds.) LATA 2013, Vol. 7810 of Lecture Notes in Computer Science, pp. 299–310. Springer, Berlin (2013)
35. Goč, D., Saari, K., Shallit, J.: Primitive words and Lyndon words in automatic and linearly recurrent sequences. In: Dediu, A.-H., Martin-Vide, C., Truthe, B. (eds.) LATA 2013, Primitive Vol. 7810 of Lecture Notes in Computer Science, pp. 311–322. Springers, Berlin (2013)
36. Goč, D., Schaeffer, L., Shallit, J.: The subword complexity of k-automatic sequences is k-synchronized. In: Béal, M.-P., Carton, O. (eds.) DLT 2013, Vol. 7907 of Lecture Notes in Computer Science, pp. 252–263. Springer, Berlin (2013)
37. Goins, E.H.: Palindromes in different bases: a conjecture of J. Ernest Wilkins. INTEGERS — Electronic J. Combinat. Number Theory **9**, 725–734 (2009). Paper #A55, (electronic)
38. Han, Y.-S., Salomaa, K.: Nondeterministic state complexity of nested word automata. Theoret. Comput. Sci. **410**, 2961–2971 (2009)
39. Haque, S., Shallit, J.: Discriminators and k-regular sequences. INTEGERS — Electronic J. Combinat. Number Theory **16**, A76 (2016). (electronic)
40. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 5th edn. Oxford University Press, Oxford (1985)
41. Harminc, M., Soták, R.: Palindromic numbers in arithmetic progressions. Fibonacci Quart. **36**, 259–262 (1998)
42. Heizmann, M., Hoenicke, J., Podelski, A.: Software model checking for people who love automata. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification — 25th International Conference, CAV 2013, vol. 8044 of Lecture Notes in Computer Science, pp. 36–52. Springer, Berlin (2013)
43. Heizmann, M., Dietsch, D., Greitschus, M., Leike, J., Musa, B., Schätzle, C., Podelski, A.: Ultimate automizer with two-track proofs. In: Chechik, M., Raskin, J.-F. (eds.) 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, Berlin (2016)
44. Hernández, S., Luca, F.: Palindromic powers. Rev. Colombiana Mat. **40**, 81–86 (2006)
45. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Boston (1979)
46. Kane, D.M., Sanna, C., Shallit, J.: Waring's Theorem for Binary Powers. arXiv:1801.04483. To appear, Combinatorica (2018)
47. Keith, M.: Classification and enumeration of palindromic squares. J. Recreat. Math. **22**(2), 124–132 (1990)
48. Kidder, P., Kwong, H.: Remarks on integer palindromes. J. Recreat. Math. **36**, 299–306 (2007)
49. Korec, I.: Palindromic squares for various number system bases. Math. Slovaca **41**, 261–276 (1991)
50. Kresová, H., Sălát, T.: On palindromic numbers. Acta Math. Univ Comenian. **42**(/43), 293–298 (1984)
51. La Torre, S., Napoli, M., Parente, M.: On the membership problem for visibly pushdown languages. In: Graf, S., Zhang, W. (eds.) ATVA 2006, vol. 4218 of Lecture Notes in Computer Science, pp. 96–109, Springer (2006)
52. Lekkerkerker, C.G.: Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci. Simon Stevin **29**, 190–195 (1952)
53. Luca, F.: Palindromes in Lucas sequences. Monatsh. Math. **138**, 209–223 (2003)
54. Luca, F., Togbé, A.: On binary palindromes of the form 10^n1. C. R. Acad. Sci. Paris **346**, 487–489 (2008)
55. Luca, F., Young, P.T.: On the Binary Expansion of the Odd Catalan Numbers. In: Proc. 14Th Int. Conf. on Fibonacci Numbers and Their Applications, pp. 185–190. Soc. Mat. Mexicana (2011)
56. Madhusudan, P., Nowotka, D., Rajasekaran, A., Shallit, J.: Lagrange's theorem for binary squares. In: Potapov, I., Spirakis, P., Worrell, J. (eds.) 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018), Leibniz International Proceedings in Informatics, pp. 18:1–18:14. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Germany (2018)
57. McCune, W.: Solution of the Robbins problem. J. Autom. Reason. **19**(3), 263–276 (1997)

58. Mehlhorn, K.: Pebbling Mountain Ranges and Its Application of DCFL-Recognition. In: Proc. 7Th Int'l Conf. on Automata, Languages, and Programming (ICALP), Vol. 85 of Lecture Notes in Computer Science, pp. 422–435. Springer (1980)
59. Mousavi, H.: Automatic theorem proving in Walnut. arXiv:1603.06017 (2016)
60. Mousavi, H., Schaeffer, L., Shallit, J.: Decision algorithms for Fibonacci-automatic words, I Basic results. RAIRO Inform. Théor. App. **50**, 39–66 (2016)
61. Nathanson, M.B.: Additive Number Theory: The Classical Bases. Springer, Berlin (1996)
62. Ogden, W.: A helpful result for proving inherent ambiguity. Math. Systems Theory **2**, 191–194 (1968)
63. Okhotin, A., Salomaa, K.: State complexity of operations on input-driven pushdown automata. J Comput. System Sci. **86**, 207–228 (2017)
64. Piao, X., Salomaa, K.: Operational state complexity of nested word automata. Theoret. Comput. Sci. **410**, 3290–3302 (2009)
65. Pollack, P.: Palindromic sums of proper divisors. INTEGERS — Electronic J. Combinat. Number Theory **15A**, Paper #A13 (electronic) (2015)
66. Rajasekaran, A.: Using automata theory to solve problems in additive number theory. Master's thesis, School of Computer Science, University of Waterloo, 2018. Available at https://uwspace.uwaterloo.ca/handle/10012/13202
67. Rajasekaran, A., Smith, T., Shallit, J.: Sums of palindromes: an approach via automata. In: Niedermeier, R., Vallée, B. (eds.) 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018), Leibniz International Proceedings in Informatics, pp. 54:1–54:12. Schloss Dagstuhl — Leibniz-Zentrum für Informatik (2018)
68. Salomaa, K.: Limitations of lower bound methods for deterministic nested word automata. Inform. Comput. **209**, 580–589 (2011)
69. Shallit, J.: Decidability and enumeration for automatic sequences: a survey. In: Bulatov, A.A., Shur, A.M. (eds.) CSR 2013, vol. 7913 of Lecture Notes in Computer Science, pp. 49–63. Springer, Berlin (2013)
70. Shorey, T.N.: On the equation $z^q = (x^n - 1)/(x - 1)$. Indag. Math. **48**, 345–351 (1986)
71. Sigg, M.: On a conjecture of John Hoffman regarding sums of palindromic numbers. arXiv:1510.07507 (2015)
72. Simmons, G.J.: On palindromic squares of non-palindromic numbers. J. Recreat. Math. **5**(1), 11–19 (1972)
73. Sloane, N.J.A. et al.: The on-line encyclopedia of integer sequences Available at https://oeis.org (2019)
74. Thue, A.: ÜBer die gegenseitige Lage gleicher Teile gewisser Zeichenreihen. Norske vid. Selsk. Skr. Mat. Nat. Kl. **1**, 1–67 (1912). Reprinted in Selected Mathematical Papers of Axel Thue, T. Nagell, editor, Universitetsforlaget, Oslo, 1977, 413–478
75. Trigg, C.: Infinite sequences of palindromic triangular numbers. Fibonacci Quart. **12**, 209–212 (1974)
76. Tymoczko, T.: The four-color problem and its philosophical significance. J. Philosophy **76**(2), 57–83 (1979)
77. Vaughan, R.C., Wooley, T.: Number Theory for the Millennium. III, pp. 301–340. A. K. Peters. In: Bennett, M.A., Berndt, B.C., Boston, N., Diamond, H.G., Hildebrand, A.J., Philipp, W. (eds.) (2002)
78. Yates, S.: The mystique of repunits. Math. Mag. **51**, 22–28 (1978)
79. Zeckendorf, E.: Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. Bull. Soc. Roy. Liége **41**, 179–182 (1972)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.