Additive Number Theory via Automata

Jeffrey Shallit
School of Computer Science, University of Waterloo
Waterloo, ON N2L 3G1 Canada
shallit@uwaterloo.ca
https://cs.uwaterloo.ca/~shallit/

Joint work with





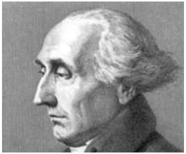

Jason Bell | Kathryn Hare | P. Madhusudan | Dirk Nowotka | Aayush Rajasekaran | Tim Smith

Additive number theory

Let S be a subset of the natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$.

The **principal problem** of additive number theory is to determine whether every natural number (or every sufficiently large natural number) can be written as the sum of some **constant** number of elements of S.

Probably the most famous example is **Lagrange's theorem** (1770):



(a) every natural number is the sum of four squares; and

(b) three squares do not suffice for numbers of the form $4^a(8k + 7)$.

(Conjectured by Bachet in 1621.)

Additive bases

Let $S \subseteq \mathbb{N}$.

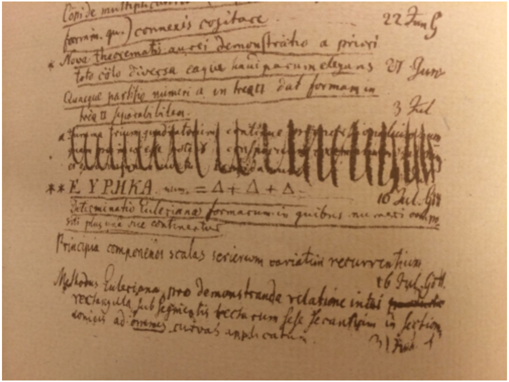
We say that a subset S is an **basis of order** h if every natural number can be written as the sum of h elements of S, not necessarily distinct.

We say that a subset S is an **asymptotic basis of order** h if every *sufficiently large* natural number can be written as the sum of h elements of S, not necessarily distinct.

Gauss's theorem for triangular numbers

A *triangular number* is a number of the form $n(n+1)/2$.

Gauss wrote the following in his diary on July 10 1796:



i.e., **The triangular numbers form an additive basis of order** 3

Waring's problem for powers

Edward Waring (1770) asserted, without proof, that every natural number is
– the sum of 4 squares
– the sum of 9 cubes
– the sum of 19 fourth powers
– "and so forth".

9. Omnis integer numerus vel eſt cubus; vel e duobus, tribus, 4, 5, 6, 7, 8, vel novem cubis compoſitus: eſt etiam quadrato-quadratus; vel e duobus, tribus, &c. uſque ad novemdecim compoſitus, & ſic deinceps: conſimilia etiam affirmari poſſunt (exceptis excipiendis) de eodem numero quantitatum earundem dimenſionum.

Waring's problem

Let $g(k)$ be the least natural number m such that every natural number is the sum of m k'th powers.

Let $G(k)$ be the least natural number m such that every sufficiently large natural number is the sum of m k'th powers.

Proving that $g(k)$ and $G(k)$ exist, and determining their values, is **Waring's problem**.

By Lagrange we know $g(2) = G(2) = 4$.

Hilbert proved in 1909 that $g(k)$ and $G(k)$ exist for all k.

By Wieferich and Kempner we know $g(3) = 9$.

We know that $4 \leq G(3) \leq 7$, but the true value is still unknown.

Other additive bases?

What other sets can be additive bases?

Not the powers of 2 – too sparse.

Need a set whose natural density is at least $N^{1/k}$ for some k.

How about numbers with palindromic base-b expansions?

Palindromes

- A *palindrome* is any string that is equal to its reversal
- Examples are `radar` (English), `ressasser` (French), and `10001`.
- We call a natural number a *base-b palindrome* if its base-b representation (without leading zeroes) is a palindrome
- Examples are $16 = [121]_3$ and $297 = [100101001]_2$.
- Binary palindromes ($b = 2$) form sequence A006995 in the *On-Line Encyclopedia of Integer Sequences* (OEIS):

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

- They have density $\Theta(N^{1/2})$.

The problem

Do the base-b palindromes form an additive basis, and if so, of what order?

William Banks (2015) showed that every natural number is the sum of at most 49 base-10 palindromes. (*INTEGERS* **16** (2016), #A3)

Javier Cilleruelo, Florian Luca, and Lewis Baxter (2017) showed that for all bases $b \geq 5$, every natural number is the sum of three base-b palindromes. (*Math. Comp.* (2017), to appear)

What we proved

However, the case of bases $b = 2, 3, 4$ was left unsolved. We proved

Theorem (Rajasekaran, JOS, Smith)

Every natural number N is the sum of 4 binary palindromes. The number 4 is optimal.

For example,

$$10011938 = 5127737 + 4851753 + 32447 + 1$$
$$= [10011100011111000111001]_2 + [10010100000100000101001]_2 +$$
$$+ [111111010111111]_2 + [1]_2$$

4 is optimal: 10011938 is not the sum of 2 binary palindromes.

Previous proofs were complicated (1)

Excerpt from Banks (2015):

2.4. **Inductive passage from** $\mathbb{N}_{\ell,k}(5^+;c_1)$ **to** $\mathbb{N}_{\ell-1,k+1}(5^+;c_2)$**.**

LEMMA 2.4. *Let* $\ell, k \in \mathbb{N}$, $\ell \geqslant k+6$, *and* $c_\ell \in \mathcal{D}$ *be given. Given* $n \in \mathbb{N}_{\ell,k}(5^+;c_1)$, *one can find digits* $a_1, \ldots, a_{18}, b_1, \ldots, b_{18} \in \mathcal{D}\backslash\{0\}$ *and* $c_2 \in \mathcal{D}$ *such that the number*

$$n - \sum_{j=1}^{18} q_{\ell-1,k}(a_j, b_j)$$

lies in the set $\mathbb{N}_{\ell-1,k+1}(5^+;c_2)$.

Proof. Fix $n \in \mathbb{N}_{\ell,k}(5^+;c_1)$, and let $\{\delta_j\}_{j=0}^{\ell-1}$ be defined as in (1.1) (with $L := \ell$). Let m be the three-digit integer formed by the first three digits of n; that is,

$$m := 100\delta_{\ell-1} + 10\delta_{\ell-2} + \delta_{\ell-3}.$$

Clearly, m is an integer in the range $500 \leqslant m \leqslant 999$, and we have

$$n = \sum_{j=k}^{\ell-1} 10^j \delta_j = 10^{\ell-3} m + \sum_{j=k}^{\ell-4} 10^j \delta_j. \tag{2.4}$$

Let us denote

$$\mathcal{S} := \{19, 29, 39, 49, 59\}.$$

In view of the fact that

$$9\mathcal{S} := \underbrace{\mathcal{S} + \cdots + \mathcal{S}}_{\text{nine copies}} = \{171, 181, 191, \ldots, 531\},$$

it is possible to find an element $h \in 9\mathcal{S}$ for which $m - 80 < 2h \leqslant m - 60$. With h fixed, let $s_1, \ldots, s_9$ be elements of $\mathcal{S}$ such that

$$s_1 + \cdots + s_9 = h.$$

Previous proofs were complicated (2)

Excerpt from Cilleruelo et al. (2017)

II.2 $\boldsymbol{c_m = 0}$. We distinguish the following cases:

II.2.i) $\boldsymbol{y_m \neq 0}$.

δ_m	δ_{m-1}
0	0
$*$	y_m
$*$	$*$

$\longrightarrow$

δ_m	δ_{m-1}
1	1
$*$	$y_m - 1$
$*$	$*$

II.2.ii) $\boldsymbol{y_m = 0}$.

II.2.ii.a) $\boldsymbol{y_{m-1} \neq 0}$.

δ_m	δ_{m-1}	δ_{m-2}
0	0	$*$
y_{m-1}	0	y_{m-1}
$*$	z_{m-1}	z_{m-1}

$\longrightarrow$

δ_m	δ_{m-1}	δ_{m-2}
1	1	$*$
$y_{m-1} - 1$	$g - 2$	$y_{m-1} - 1$
$*$	$z_{m-1} + 1$	$z_{m-1} + 1$

The above step is justified for $z_{m-1} \neq g - 1$. But if $z_{m-1} = g - 1$, then $c_{m-1} \geq (y_{m-1} + z_{m-1})/g \geq 1$, so $c_m = (z_{m-1} + c_{m-1})/g = (g - 1 + 1)/g = 1$, a contradiction.

II.2.ii.b) $\boldsymbol{y_{m-1} = 0,\ z_{m-1} \neq 0}$.

δ_m	δ_{m-1}	δ_{m-2}
0	0	$*$
0	0	0
$*$	z_{m-1}	z_{m-1}

$\longrightarrow$

δ_m	δ_{m-1}	δ_{m-2}
0	0	$*$
1	1	1
$*$	$z_{m-1} - 1$	$z_{m-1} - 1$

II.2.ii.c) $\boldsymbol{y_{m-1} = 0,\ z_{m-1} = 0}$.

If also $c_{m-1} = 0$, then $\delta_{m-1} = 0$, which is not allowed. Thus, $c_{m-1} = 1$.

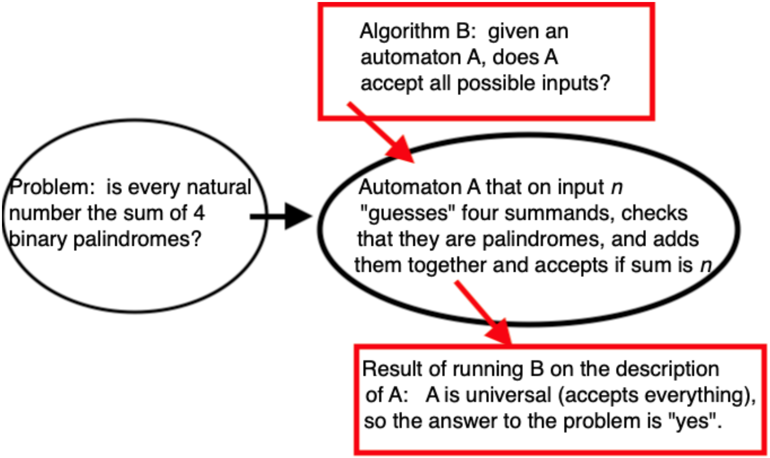
Previous proofs were complicated (3)

- Proofs of Banks and Cilleruelo et al. were long and case-based
- Difficult to establish
- Difficult to understand
- Difficult to check, too: the original Cilleruelo et al. proof had some minor flaws that were only noticed when the proof was implemented as a `Python` program
- Idea: could we automate such proofs?

The main idea of our proof

- Construct a finite-state machine (automaton) that takes natural numbers as input, expressed in the desired base
- Allow the automaton to nondeterministically “guess” a representation of the input as a sum of palindromes
- The machine accepts an input if it “verifies” its guess
- Then use a decision procedure to establish properties about the language of representations accepted by this machine (e.g., universality – does it accept every possible input?)

Our proof strategy



Basics of automata

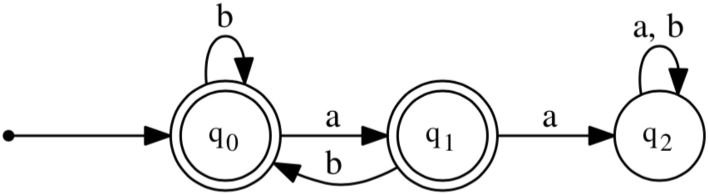
- An automaton is a mathematical model of a very simple computer
- It takes as input a finite list of symbols $x = a_1 a_2 \cdots a_n$, called a "string" or "word")
- The automaton does some computation and then either "accepts" or "rejects" its input
- The set of all accepted strings is called the *language recognized* by the automaton

Parts of an automaton

- The finite set of *states*: each state corresponds to some knowledge that has been gained about the input
- The start state
- The set of accepting states
- The transition function that specifies, for each state and each input symbol, which state to enter

Example of an automaton

A double circle represents an accepting state.

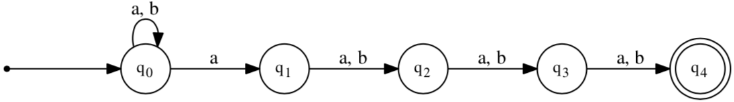


What is the language accepted by this automaton?

It is the set of all strings having no two consecutive a's.

Different kinds of automata

- Some have extra storage, in the form of a stack ("last in, first out"); they are called "pushdown automata"
- One very powerful tool: nondeterminism
- Here the automaton is allowed to "guess" what moves to make, and then "verify" that its guess is correct
- Example: accept those strings where the 4th symbol from the end is an a



Decision algorithms for automata

- Given an automaton, we can decide various things about the language it recognizes.
- For example, is the language empty? Or infinite?
- Here "decide" means there is an algorithm that, given the automaton as input, halts and says (for example) either "language is empty" or "language is not empty".
- In some cases, we can also decide universality: the property of accepting all strings.

Picking an automaton for palindromes

What kind of automaton should we choose?

- it should be possible to check if the guessed summands are palindromes
  - can be done with a pushdown automaton (PDA)
- it should be possible to add the summands and compare to the input
  - can be done with a finite automaton (DFA or NFA)

However

- Can't add summands with these machine models unless they are guessed in parallel
- Can't check if summands are palindromes if they are wildly different in length & presented in parallel
- Universality is not decidable for PDA's

What to do?

Visibly pushdown automata (VPA)

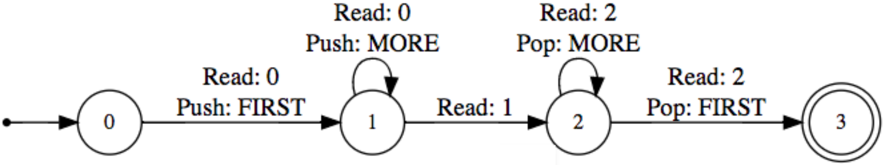
- Use *visibly-pushdown automata*!
- Popularized by Alur and Madhusudan in 2004, though similar ideas have been around for longer
- VPA's receive an input string, and read the string one letter at a time
- They have a (finite) set of states and a stack
- Upon reading a letter of the input string, the VPA can transition to a new state, and might modify the stack

Using the VPA's stack

- The VPA can only take very specific stack actions
- The input alphabet, Σ, is partitioned into three disjoint sets
  - Σ_c, the push alphabet
  - Σ_l, the local alphabet
  - Σ_r, the pop alphabet
- If the letter of the input string we read is from the push alphabet, the VPA pushes *exactly* one symbol onto its stack
- If the letter of the input string we read is from the pop alphabet, the VPA pops *exactly* one symbol off its stack
- If the letter of the input string we read is from the local alphabet, the VPA does not consult its stack at all

Example VPA

A VPA for the language $\{0^n 1 2^n \ : \ n \geq 1\}$:



The push alphabet is $\{0\}$, the local alphabet is $\{1\}$, and the pop alphabet is $\{2\}$.

Determinisation and Decidability

- A nondeterministic VPA can have several matching transition rules for a single input letter
- Nondeterministic VPA's are as powerful as deterministic VPA's
- VPL's are closed under union, intersection and complement. There are algorithms for all these operations.
- Testing emptiness, universality and language inclusion are decidable problems for VPA's
- But a nondeterministic VPA with n states can have as many as $2^{\Theta(n^2)}$ states when determinized!

Proof strategy

- We build a VPA that nondeterministically "guesses" strings representing integers that are of roughly the same size, in parallel
- It checks to see that the guesses are palindromes
- It adds the guessed numbers together and verifies that the sum equals the input number.
- There are some complications due to the VPA restrictions.

More details of the proof strategy

- To prove our result, we built 2 VPA's A and B:
  - A accepts all n-bit odd integers, $n \geq 8$, that are the sum of three binary palindromes of length either
    - n, $n-2$, $n-3$, or
    - $n-1$, $n-2$, $n-3$.
  - B accepts all valid representations of odd integers of length $n \geq 8$
- We then prove that all inputs accepted by B are accepted by A
- We used the ULTIMATE Automata Library
- Once A and B are built, we simply have to issue the command

$$\texttt{assert(IsIncluded(B, A))}$$

in ULTIMATE.

Finishing up the proof

- Thus every odd integer ≥ 256 is the sum of three binary palindromes.
- For even integers, we just include 1 as one of the summands.
- The numbers < 256 are easily checked by brute force.
- And so we've proved: every natural number is the sum of four binary palindromes.

Bases 3 and 4

- Unfortunately, the VPA's for bases 3 and 4 are too large to handle in this way.
- So we need a different approach.
- Instead, we use ordinary nondeterministic finite automata (NFA).
- But they cannot recognize palindromes...
- Instead, we change the input representation so that numbers are represented in a "folded" way, where each digit at the beginning of its representation is paired with its corresponding digit at the end.
- With this we can prove...

Other results

Theorem

Every natural number $N > 256$ is the sum of at most three base-3 palindromes.

Theorem

Every natural number $N > 64$ is the sum of at most three base-4 palindromes.

This completes the classification for base-b palindromes for all $b \geq 2$.

More results

Using NFA's we can establish an analogue of Lagrange's four-square theorem.

- A *square* is any string that is some shorter string repeated twice
- Examples are `hotshots` (English), `chercher` (French), and 100100.
- We call an integer a *base-b square* if its base-b representation is a square
- Examples are $36 = [100100]_2$ and $3 = [11]_2$.
- The binary squares form sequence A020330 in the OEIS

$$3, 10, 15, 36, 45, 54, 63, 136, 153, 170, 187, 204, 221, \ldots$$

Results

Theorem

Every natural number $N > 686$ is the sum of at most 4 binary squares.

For example:

$$\begin{aligned} 10011938 &= 9291996 + 673425 + 46517 \\ &= [100011011100100011011100]_2 + [10100100011010010001]_2 \\ &+ [1011010110110101]_2 \end{aligned}$$

We also have the following result

Theorem

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

Generalizing: Waring's theorem for binary k'th powers

Recall Waring's theorem: *for every $k \geq 1$ there exists a constant $g(k)$ such that every natural number is the sum of $g(k)$ k'th powers of natural numbers.*

Could the same theorem hold for binary k'th powers?

Two issues:

- 1 is not a binary k'th power, so it has to be "every sufficiently large natural number" and not "every natural number".
- The gcd g of the binary k'th powers need not be 1, so it actually has to be "every sufficiently large multiple of g".

gcd of the binary k'th powers

Theorem

The gcd *of the binary k'th powers is* $\gcd(k, 2^k - 1)$.

Example:

The binary 6'th powers are

$$63, 2730, 4095, 149796, 187245, 224694, 262143, 8947848, 10066329, \ldots$$

with gcd equal to $\gcd(6, 63) = 3$.

Very recent results

Theorem

Every sufficiently large multiple of $\gcd(k, 2^k - 1)$ *is the sum of a constant number (depending on* k*) of binary* k*'th powers.*

Obtained with Daniel Kane and Carlo Sanna.

Outline of the proof

Given a number N we wish to represent as a sum of binary k'th powers:

- choose a suitable power of 2, say 2^n, and express N in base 2^n.
- use linear algebra to change the basis and instead express N as a linear combination of $c_k(n), c_k(n+1), \ldots, c_k(n+k-1)$ where

$$c_k(n) = \frac{2^{kn} - 1}{2^n - 1}.$$

- Such a linear combination would seem to provide an expression for N in terms of binary k'th powers, but there are three problems to overcome:
  - (a) the coefficients of $c_k(i)$, $n \leq i < n + k$, could be much too large;
  - (b) the coefficients could be too small or negative;
  - (c) the coefficients might not be integers.

All of these problems can be handled with some work.

Other results

Call a set S of natural numbers *b-automatic* if the language of the base-b expansions of its members is regular (accepted by a finite automaton).

Theorem (Bell, Hare, JOS)

It is decidable, given a b-automatic set S, whether it forms an additive basis (resp., asymptotic additive basis) of finite order.
If it does, the minimum order is also computable.

The proof uses, in part, a decidable extension of Presburger arithmetic.

An Open Problem

How many states are needed, in the worst case, for an automaton to accept one specified string w of length n, but reject another string x of the same length?

Best lower bound known: in some cases $\Omega(\log n)$ states are needed.

Best upper bound known: in all cases $O(n^{2/5}(\log n)^{3/5})$ states suffice.

These are widely separated!

I offer CDN \$ 200 for a solution to this problem.

For further reading

1. A. Rajasekaran, J. Shallit, T. Smith, Additive number theory via automata theory, *Theor. Comput. Systems* (2019). Available at https://doi.org/10.1007/s00224-019-09929-9.
2. P. Madhusudan, D. Nowotka, A. Rajasekaran, and J. Shallit, Lagrange's theorem for binary squares, 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018), Article No. 18, pp. 18:1–18:14, Leibniz International Proceedings in Informatics, 2018.
3. J. Bell, K. Hare, and J. Shallit, When is an automatic set an additive basis? *Proc. Amer. Math. Soc. Ser. B* **5** (2018), 50–63.
4. D. M. Kane, C. Sanna, and Jeffrey Shallit, Waring's theorem for binary powers, to appear, *Combinatorica*.