Experimental Combinatorics on Words Using the Walnut Prover

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

Joint work with Jean-Paul Allouche, Émilie Charlier, Narad Rampersad, Dane Henshall, Luke Schaeffer, Eric Rowland, Daniel Goč, and Hamoon Mousavi.

What is Combinatorics on Words?

1. The study of the properties of finite and infinite words (strings of symbols) over a finite alphabet Σ
2. For example, the famous Lyndon-Schützenberger theorem describes when the product (concatenation) of two words commutes: when $xy = yx$
3. The Fine-Wilf theorem describes how long two infinite periodic sequences, of period h and k, can agree — without agreeing forever

Seven Points of this Talk

1. Experimental techniques can be used to guess infinite words satisfying a given prefix-invariant property P
2. Once the answer has been guessed, it can often be stated in first-order logic in an extension of Presburger arithmetic
3. An automaton-based decision procedure exists for many such extensions
4. The decision procedure is relatively easy to implement and often runs remarkably quickly, despite its formidable worst-case complexity — and we have an implementation that is publicly available (`Walnut`)

Seven Points of the Talk

5. Many results already in the literature (in dozens of papers and Ph. D. theses) can be reproved by our program in a matter of seconds (including fixing at least one that was wrong!)
6. Many new results can be proved
7. There are some well-defined limits to what we can do because either
    - the property is not expressible in first-order logic; or
    - the underlying sequence leads to undecidability

A classical avoidability problem in words

- A *square* is a nonempty block of the form xx, where x is a word.
- Examples in English include `hotshots` and `murmur`
- It is easy to see that every word of length ≥ 4 over a 2-letter alphabet has a square within it: either 00 or 11 or 0101 or 1010.
- But how about words over a 3-letter alphabet?
- Thue proved that the infinite word

  $$\mathbf{a} = a_0 a_1 a_2 \cdots = 210201 \cdots ,$$

  generated by iterating the morphism $2 \to 210$, $1 \to 20$, and $0 \to 1$, is **squarefree**.
- Once guessed, this result can be rigorously proved using our decision procedure.

A classical avoidability problem in words

- We hope that **a** is an **automatic sequence**, that is, it is generated by a finite automaton as follows:
  - The automaton must accept inputs $n \geq 0$ represented in some base k, and reach a state with associated output a_n
- It turns out that base-2 works with the following automaton:

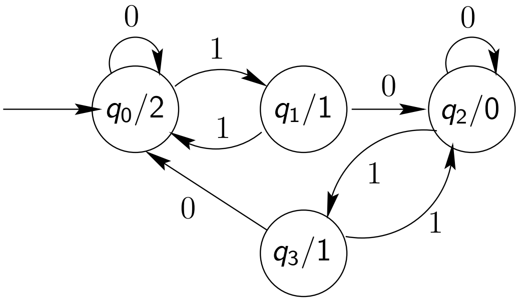

Figure : The automaton for Thue's sequence

Using the Walnut prover

To use the Walnut prover, first we define the automaton TH in a file called TH.txt. Then we run the prover. Here's the output:

```
eval thues "Ei En (n>=1) & Aj (j<n) => TH[i+j]=TH[i+j+n]":
n>=1 has 2 states: 107ms
j<n has 2 states: 1ms
TH[(i+j)]=TH[((i+j)+n)] has 12 states: 169ms
(j<n=>TH[(i+j)]=TH[((i+j)+n)]) has 25 states: 20ms
(A j (j<n=>TH[(i+j)]=TH[((i+j)+n)])) has 1 states: 215ms
(n>=1 & (A j (j<n=>TH[(i+j)]=TH[((i+j)+n)]))) has 1 states: 2ms
(E n (n>=1 & (A j (j<n=>TH[(i+j)]=TH[((i+j)+n)])))) has 1 states: 1ms
(E i (E n (n>=1 & (A j (j<n=>TH[(i+j)]=TH[((i+j)+n)]))))) has 1 states: 1ms
total computation time: 578ms
```

and the output is "false".

A general approach to finding infinite sequences satisfying a prefix-invariant property

- There is a heuristic method to find infinite sequences satisfying some prefix-invariant property P, similar to what we did for avoiding squares.
- If the method succeeds, it actually provides a proof of correctness.
- The method is to guess an appropriate automaton and then verify its correctness using our prover.
- There are two things left to explain:
  1. How do we guess the automaton, if it exists?
  2. How does the prover work?

If the sequence can be computed

If the sequence can be explicitly computed and there is an automaton calculating it, we can use a decimation procedure to guess the automaton:

- We start by taking the sequence and “decimating” it; that is, we form a new sequence by taking every k'th term starting with a_0, then every k'th term starting with a_1, and so forth, up to starting with a_{k-1}
- This gives us k subsequences:

$$
\begin{aligned}
&a_0 a_k a_{2k} \cdots \\
&a_1 a_{k+1} a_{2k+1} \cdots \\
&\vdots \\
&a_{k-1} a_{2k-1} a_{3k-1} \cdots
\end{aligned}
$$

If the sequence can be computed

- We then try to match these sequences against previously computed subsequences of the original sequence; if two agree on hundreds or thousands of terms, we guess that they agree forever
- Any unmatched sequence is then decimated in the same way, until no unmatched sequences remain.
- From this we can make an automaton, with each sequence represented by a state

If the sequence is unknown

If the sequence satisfying the property P is unknown

- We can use breadth-first search to enumerate all strings w of length $1, 2, 3, \ldots$ satisfying P
- For each string we can efficiently find the minimal automaton generating an infinite sequence for which w is a prefix
- We can then use our decision procedure on this automaton
- If an automaton generating a sequence with property P, this will eventually find it

The procedure with the Thue sequence

```
a[n] = 210201210120210201202101210201210120210121020120210201210120210201202101
a[2n] = 20020220022020020220200220020220022020022002022020020220022020020220200
a[2n+1] = 121110121011121110111210121110121011121012111011121110121011121110111
a[4n] = a[2n]
a[4n+1] = 111111111111111111111111111111111111111111111111111111111111111111111
a[4n+2] = 022020022002022020020220022020022002022002202002022020022002022020020
a[4n+3] = a[n]
a[8n+1] = a[4n+1]
a[8n+2] = a[4n+2]
a[8n+5] = a[4n+1]
a[8n+6] = a[2n]
```

From these sequences we can form the automaton which accepts the sequence in lsd format.

The lsd-first automaton for the Thue sequence

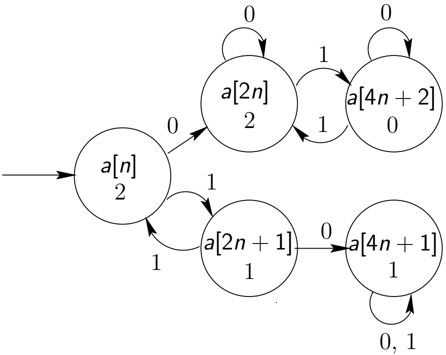

Figure : The lsd-first automaton for Thue's sequence

Now a standard technique for reversing the digits in an automaton gives us the automaton we saw before.

First-order logic

- Let $\mathrm{Th}(\mathbb{N}, +, 0, 1)$ denote the set of all true first-order sentences in the logical theory of the natural numbers with addition.
- This is sometimes called *Presburger arithmetic*.
- Here we are allowed to use any number of variables, logical connectives like "and", "or", "not", etc., and quantifiers like $\exists$ and $\forall$.

Presburger's theorem

Figure : Mojżesz Presburger (1904–1943)

Presburger proved that $\mathrm{Th}(\mathbb{N}, +, 0, 1)$ is *decidable*: that is, there exists an algorithm that, given a sentence in the theory, will decide its truth. He used quantifier elimination.

Decidability of Presburger arithmetic: Rabin's proof

Rabin found a much simpler proof of Presburger's result, based on automata.

Ideas:

- represent integers in an integer base $k \geq 2$ using the alphabet $\Sigma_k = \{0, 1, \ldots, k-1\}$.
- represent n-tuples of integers as words over the alphabet Σ_k^n, padding with leading zeroes, if necessary. This corresponds to reading the base-k representations of the n-tuples *in parallel*.
- For example, the pair $(21, 7)$ can be represented in base 2 by the word

$$[1, 0][0, 0][1, 1][0, 1][1, 1].$$

Decidability of predicates

The relation $x + y = z$ can be checked by a simple 2-state automaton depicted below, where transitions not depicted lead to a nonaccepting "dead state".

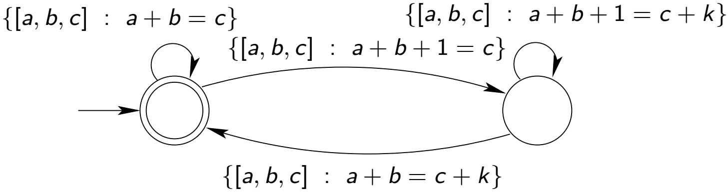

Figure : Checking addition in base k

Decidability of Presburger arithmetic: proof sketch

- Relations like $x = y$ and $x < y$ can be checked similarly.
- Given a formula with free variables $x_1, x_2, \ldots, x_n$, we construct an automaton accepting the base-k expansion of those n-tuples $(x_1, \ldots, x_n)$ for which the proposition holds.
- If a formula is of the form $\exists x_1, x_2, \ldots x_n \; p(x_1, \ldots, x_n)$, then we use nondeterminism to "guess" the x_i and check them.
- If the formula is of the form $\forall p$, we use the equivalence $\forall p \equiv \neg \exists \neg p$; this may require using the subset construction to convert an NFA to a DFA and then flipping the "finality" of states.
- Finally, the truth of a formula can be checked by using the usual depth-first search techniques to see if any final state is reachable from the start state.

The bad news

- The worst-case running time of the algorithm above is bounded above by

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{p(N)}}}}}},$$

  where the number of 2's in the exponent is equal to the number of quantifier alternations, p is a polynomial, and N is the number of states needed to describe the underlying automatic sequence.
- This bound can be improved to double-exponential.

The good news

- With a small extension to Presburger's logical theory — adding the function $V_k(n)$, the largest power of k dividing n — one can also verify many more interesting statements (examples to follow). But then the worst-case time bound returns to

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{p(N)}}}}}}.$$

- Beautiful theory due to Büchi, Bruyère, Hansel, Michaux, Villemaire, etc.
- Despite the awful worst-case bound on running time, an implementation often succeeds in verifying statements in the theory in a reasonable amount of time and space.
- Many old results from the literature can be verified with this technique, and many new ones can be proved.

An extended example: avoiding the pattern xxx^R

- By x^R we mean the reversal of the string x. For example, $(\texttt{stressed})^R = \texttt{desserts}$.
- An example of this pattern in English is contained in the word

  $\texttt{bepepper}$.
- Are there infinite binary words avoiding this pattern?

An extended example: avoiding the pattern xxx^R

- We start by trying depth-first search.
- It gives the lexicographically least such sequence.
- This gives the word

$$(001)^3(10)^\omega = 001001001101010 \cdots .$$

- So in particular the word $(10)^\omega = 101010 \cdots$ avoids the pattern. (Easy proof!)
- This suggests: are there any other periodic infinite words avoiding xxx^R?
- Also: are there any aperiodic infinite words avoiding xxx^R?

An extended example: avoiding the pattern xxx^R

When we search for other primitive words z such that z^ω avoids the pattern, we find there are some of length 10:

0010011011 0011011001 0100110110 0110010011 0110110010
1001001101 1001101100 1011001001 1100100110 1101100100

- We notice that each of these words is of the form $w\overline{w}$.
- This suggests looking at words of this form.
- The next ones are $w = 0010010011011001001 00$, and its shifts and complements.

An extended example: avoiding the pattern xxx^R

- To summarize, here are the solutions we've found so far:

w	$\|w\|$
01	2
00100	5
001001001101100100100	21

- The presence of the numbers 2,5,21 suggests some connection with the Fibonacci numbers.

An aperiodic word avoiding xxx^R

- Suppose we take the run-length encodings of the strings of length 21. One of them looks familiar: 2122121221221. This is a prefix of the infinite Fibonacci word generated by $2 \to 21$, $1 \to 2$.
- This suggests the construction of an *infinite* aperiodic word avoiding xxx^R: take the infinite Fibonacci word, and use it as "repetition factors" for 0 and 1 alternating. This gives the word

$$\mathbf{R} = 001001101101100100110 \cdots$$

  which we conjecture avoids xxx^R.
- Can we find an automaton generating this sequence? Yes, but now it is not based on base-2 representations, but rather Fibonacci (or "Zeckendorf") representations.

An aperiodic word avoiding xxx^R

- Every non-negative integer can be represented, essentially uniquely, as a sum of distinct Fibonacci numbers, provided that we never use two adjacent Fibonacci numbers.
- We can try to find an automaton for our sequence in much the same way as we did for Thue's sequence.
- When we do, we get the following automaton of 8 states.

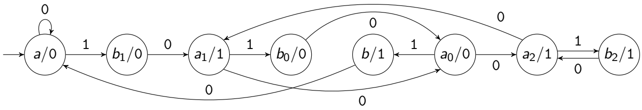

Figure : Fibonacci automaton generating the sequence **R**

An aperiodic word avoiding xxx^R

- We now have the conjecture that the word generated by this automaton (a) is aperiodic and (b) avoids xxx^R.
- Both conjectures can be proved using our decision procedure.
- We just need to write predicates for them:
  - Ultimate periodicity:

$$\exists p \geq 1 \; \exists N \geq 0 \; \forall i \geq N \; \mathbf{R}[i] = \mathbf{R}[i+p].$$

  - Has xxx^R:

$$\exists i \geq 0 \; \exists n \geq 1 \; \forall t < n$$
$$(\mathbf{R}[i+t] = \mathbf{R}[i+t+n]) \;\wedge\; (\mathbf{R}[i+t] = \mathbf{R}[i+3n-1-t]).$$

What other properties of automatic sequences are decidable?

- A difficult candidate: abelian properties
- We say that a nonempty word x is an *abelian square* if it is of the form ww' with $|w| = |w'|$ and w' a permutation of w. (An example in English is the word `reappear`.)
- Luke Schaeffer showed that the predicate for abelian squarefreeness is indeed inexpressible in $\mathrm{Th}(\mathbb{N}, +, 0, 1, V_k)$
- However, for some sequences (e.g., Thue-Morse, Fibonacci) many abelian properties are decidable

Other limits to the approach

- Consider the morphism $a \to abcc$, $b \to bcc$, $c \to c$.
- The fixed point of this morphism is

$$\mathbf{s} = abccbccccbccccccbccccccccb \cdots$$

- It encodes, in the positions of the b's, the characteristic sequence of the squares.
- So the first-order theory $\mathrm{Th}(\mathbb{N}, +, 0, 1, n \to \mathbf{s}[n])$ is powerful enough to express the assertion that "n is a square"
- With that, one can express multiplication, and so it is undecidable (Matiyasevich).

Two Open Problems

- Let p denote the characteristic sequence of the prime numbers. Is the logical theory $\mathrm{Th}(\mathbb{N}, +, 0, 1, n \rightarrow p(n))$ decidable?
- Is the following problem decidable? Given two k-automatic sequences $(a(n))_{n \geq 0}$ and $(b(n))_{n \geq 0}$, are there integers $c \geq 1$ and $d \geq 0$ such that $a(n) = b(cn + d)$ for all n?

The Walnut Prover

Our publicly-available prover, written by Hamoon Mousavi, is called `Walnut` and can be downloaded from

`www.cs.uwaterloo.ca/~shallit/papers.html` .