Review of

The Logical Approach to Automatic Sequences: Exploring Combinatorics on Words with `Walnut`

by Jeffrey Shallit

Publisher: Cambridge University Press, 2023

$70.00 Paperback, $66.00 Kindle, 360 pages

Reviewer: William Gasarch (gasarch@umd.edu)

1 Introduction

This book is concerned with the following question:

Given a sequence s does it have property p?

That is a rather broad question. What kind of sequences are allowed? How are they to be given? What kind of property is allowed? How are they given?

Here is an example of a question they consider.

Sequence Let

$X_1 = 1$

$X_2 = 0$

For all $i \geq 3$, $X_i = X_{i-1}X_{i-2}$ (concatenation).

The *Infinite Fibonacci Sequence* is $\lim_{i\to\infty} X_i$.

Question Is the following true: For every n there is a palindrome of length n that is a subword of the Infinite Fibonacci Sequence.

You might approach this by generating by hand (say) the first 20 bits. Or perhaps write a program to generate the first (say) 100 bits. And then see how it looks. You might write a program that generates the first (say) 1000 bits and then tries to find palindromic subwords of length $1, 2, 3, \ldots$ and see how far you get. These would all be aiming at a human-readable proof where the computer *helps you* to find the answer, and perhaps the proof.

Or you might just ask some program for the answer directly.

Really? Can we do that? Is there an algorithm to do that? Is it fast enough to be worth coding up? Did someone code it up? Is there a user's manual for it? Is there a book that ties all of this together so it tells you what such a package can do, why it works, and how to use it? The answers are yes, yes, yes, yes, and yes.

Véronique Bruyère et al. [2] describe an algorithm for this problem. The algorithm is fast enough so that it could be coded up. Hamoon Mousavi [4] has coded it up in a package called `Walnut` which is available. The reference given is to a user's manual. The book under review ties it all together.

More precisely, `Walnut` is a computer package that will, given a sequence (we will say what kind of sequence and how it is given later) and a question about that sequence (we will say what kind of question and how it is given later) outputs the answer to that question.

This book does the following:

1. Gives many sequences, questions about sequences, and the answers.

2. Describes the theory behind `Walnut` including the proofs of decidability when needed.

3. Instructs the reader on how to use `Walnut`. The reader can then use `Walnut` and investigate questions about sequences.

2 What Kind of Sequences Can You Ask About?

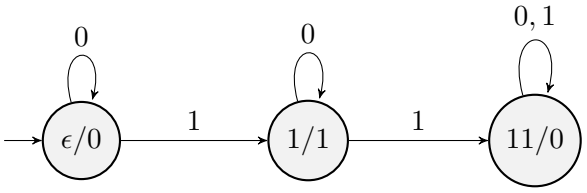
We first define a variant of DFAs that also has an output.
Definition A *Deterministic Finite Automata with Output (DFAO)* is a tuple $(Q, \Sigma, \Gamma, s, \delta, \tau)$ such that

1. Q is a finite set of states.
2. Σ and Γ are finite alphabets.
3. $s \in Q$.
4. $\delta : Q \times \Sigma \to Q$.
5. $\tau : Q \to \Gamma$
6. If $x \in \Sigma^*$ then we can run it through the DFAO in the usual way and get to a state q. Note that we do not have a notion of accept state. Instead we think of x as being mapped to $\tau(q)$. Hence a DFAO computes a function from Σ^* to Γ.
7. Let $k \in \mathsf{N}$. If $\Sigma = \{0, \ldots, k-1\}$ then we interpret the input as a base-k number and hence the DFAO can be identified with a function from N to Γ. (Alternatively the DFAO an be identified with an infinite sequence of elements of Γ.) We call such a DFAO a *k-DFAO*.

We now define the type of sequence that `Walnut` can be asked about.
Definition Let Γ be a finite alphabet. A *k-automatic sequence* is an element $x \in \Gamma^\omega$ such that there is a k-DFAO that, on input n (a base k number) ends in a state labelled $x(n)$.
Example Let x be the characteristic sequence of the powers of 2. So the sequence begins

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	1

The following is a 2-DFAO for this sequence. Each state has the name of the state (one of $\{\epsilon, 1, 11\}$) and then the label (one of $\{0, 1\}$). On input a number n in base 2, the DFAO ends in a state labelled 1 iff the nth bit of the sequence is 1 iff n is a power of 2.



Recall that `Walnut` takes as input a sequence and question about that sequence. The way to input a sequence is by inputting the DFAO for it.

3 What Kinds of Questions Can You Ask?

We define an extension of Presburger Arithmetic.
Definition We define formulas inductively. Note that there are two kinds of variables.

1. A *term* is either (1) a constant $c \in \mathsf{N}$, (2) a variable a (the domain of a is N), (3) if s, t are terms then $s + t$ is a term. (4) $x[t]$ where t is a term (x represents a sequence of natural numbers, so for example this could be $x[3]$ or $x[i]$ or $x[i + j]$). Terms 1,2,3 are in Presburger arithmetic. Term 4 is the extension.

2. If s, t are terms then $s = t$ and $s < t$ are atomic formulas.

3. If $\phi_1(x_1, \ldots, x_n)$ and $\psi(y_1, \ldots, y_m)$ are formulas then the following are formulas:

   (a) $\phi_1(x_1, \ldots, x_n) \vee \psi(y_1, \ldots, y_m)$. (We can write $\wedge$ by using De Morgan's law.)

   (b) $\neg\phi_1(x_1, \ldots, x_n)$.

   (c) For $1 \leq i \leq n$, $(\exists x_i)[\phi_1(x_1, \ldots, x_n)]$. (We can write $\forall$ by using $\forall = \neg\exists\neg$.)

4. A *Sentence* is a formula with no free variables.

5. A *Question* is a formula with only one variable of sequence type, and that variable is free, together with what sequence you want. For example if you want to know if the Thue-Moore Sequence is cube-free you would have the question with x as the sequence, and another place where you would specify that x is the Thue-Moore Sequence.

Note We have omitted other extensions where additional functions are added.
Examples of Questions

1. $(\forall i)(\exists j)[(j > i) \wedge x[i] = 0]$.

   This question asks if the sequence x is 0 infinitely often.

2. $(\exists n)(\exists i)[n \geq 1 \wedge [(\forall j)[j < n \rightarrow x[i + j] = x[i + j + n]]]$.

   This question asks if the sequence x has a square, which is a subword of the form yy.

The following Theorem is the key to `Walnut`.
Theorem The following problem is decidable: Given a sequence via the DFAO (the alphabet is base-k numbers for some k) and a question, determine if the answer to the question is yes.

4 Pros and Cons of This Approach to Mathematics

Hilbert (in today's language) wanted an algorithm that would, given a problem in math, output the answer. Gödel showed this could not be done. However, there were some subsets of math that were decidable (e.g., Presburger arithmetic). Are these systems useful for finding out answers to math problems? Before reading this book I would have answered No and made the following points:

1. You cannot state anything of interest in these systems. I give one exception; however, my second point will negate it. Rabin showed S2S is decidable [5] (see Gurevich and Harrington [3] or the book by Börger, Erich Grädel, and Yuri Gurevich [1] for an easier proof). Rabin points out that the theorem of Wolfe, that every Gale-Stewart Game with a set in F_σ is determiniate, is expressible in S2S. This theorem came out around 10 years before Rabin's paper (there are a few other math problems of interest that can be stated in S2S). If Wolfe's result was still an open problem then could the decidability of S2S have solved it?

2. Many theories that are decidable have terrible run times and are quite complicated to code up. I doubt that S2S has ever been coded up.

3. One counterthought: The theory WS2S has been coded up and has been used to verify hardware and low-level code. See here: `https://www.brics.dk/mona/`. While this is of course interesting, its not proving *mathematical* theorems of interest.

`Walnut` is a strong counterexample to the notion that decidable theories cannot be used to prove real theorems. As such it provides *two* uses in a course in Automata theory: (1) the obvious one, the students can use it to ask questions about sequences, and (2) the teacher can point to it as a decidable theory that can actually be used.

Having said this, I now list PROS and CONS to using `Walnut`.

PRO: We can get answers to questions of interest.

CON: Do these proofs give us insights into what is really going on? (I do not have the intuition for why, for every n, the infinite Fib number sequence has a palindromic subword of length n).

COUNTER: Once you know a statement is true then you may be able to find a human-readable proof.

PRO: There are times when you really just want the answer. I am thinking of proving that competing actors for a resource all get served (version of the dining philosophers problem). I wonder if `Walnut` could be used on such problems.

PRO: Speculation: Is `Walnut` doing the dreary part of the proof freeing up us humans to do other things?

CON: What if we had a program that could tell us if P=NP and we asked it and it said P $\neq$ NP. Well... we already knew that. Are we enlightened?

5 Opinion

You should buy this book for the following reasons.

1. For your own enlightenment.

2. For examples of concepts you can teach in a course in automata theory. That is, even if you do not plan to use or teach `Walnut`, there is much of interest in the book.

References

[1] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer, 1997.

[2] V. Bruyère, G. Hansel, C. Michaux, and R. Villemaire. Logic and p-recognizable sets of integers. *Bulletin of the Belgian Math. Soc.– Simon Stevin*, 1(2):191–238, 1994. Corrigendum 1:577, 1994.

[3] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing,* San Francisco CA, pages 60–65, 1982.

[4] H. Mousavi. Automatic theorem proving in Walnut, 2016. `https://arxiv.org/abs/1603.06017`.

[5] M. Rabin. Decidabilty of second-order theories of automta on infinite trees. *Transactions of the American Math Society*, 141:1–35, 1969. `www.jstor.org/stable/1995086`.