State Complexity of Shifts of the Fibonacci Word

Delaram Moradi, Pierre Popoli, Jeffrey Shallit, Ingrid Vukusic

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

State complexity

The *state complexity* of a regular language L is the number of states in the minimal deterministic finite automaton (DFA) recognizing L.
We can study (just to name two things)

- The state complexity of a parameterized set of languages L_n; or
- The worst-case state complexity of an operation on regular languages, such as union, concatenation, or Kleene star.

Pioneered by Aleksandr Nikolaevich Maslov (1947–2025) in the former Soviet Union (early 1970's) and Albert Meyer and Michael Fischer (1971) in the USA. Later popularized in the West by a 1994 paper of Sheng Yu, Qingyu Zhuang, and Kai Salomaa.

A. N. Maslov (1947–2025)

Automatic sequences

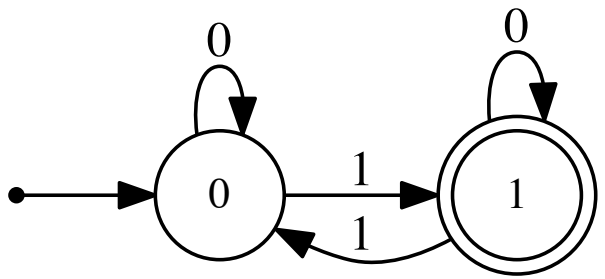
A sequence $(a_i)_{i \geq 0}$ over a finite alphabet is said to be *automatic* if there is a deterministic finite automaton with outputs associated with the states (DFAO) that, on input i represented in some numeration system (e.g., base $k \geq 2$), ends in a state with output a_i.

First studied by Alan Cobham starting about 1968.

Canonical example is the Thue-Morse sequence

$$\mathbf{t} = 0110100110010110\cdots$$

generated by the automaton at right.



Automaton generating the Thue-Morse sequence

Combining these two subjects

We can combine these two subjects, and study the **state complexity of various natural operations on automatic sequences.**

For example, the linearly-indexed subsequence of an automatic sequence is also automatic, so we can study the state complexity of linearly-indexed subsequences of an automatic sequence $(a_{Ni})_{i\geq 0}$, as a function of N.

This problem was studied by Zantema and Bosma, and more recently by Moradi et al., and Narad Rampersad gave a talk about it at the CIAA conference earlier this week.

Sample result: for the Thue-Morse sequence the state complexity of $(t_{Nn})_{n\geq 0}$ is $\Theta(N)$ for odd N. More precisely, for odd N it is exactly the same as the subword complexity of the Thue-Morse sequence at N.

Why I like this area

It involves many of my favorite things:

- number theory (continued fractions, Diophantine approximation)
- state complexity
- combinatorics on words
- first-order logic and its power
- dynamical systems

Shifts of automatic sequences

The shift by c of an automatic sequence $(a_i)_{i\geq 0}$ is also automatic.

So we can study the **state complexity of the shifted sequence** $(a_{i+c})_{i\geq 0}$ as a function of c.

In general, the answer can depend on whether the representation of the input i to the automaton is given with most-significant digit first (msd-first) or least-significant digit first (lsd-first).

For the Thue-Morse sequence and msd-first this is a surprisingly subtle question, and the exact answer is not known. It is easy to prove it is $O(c)$. The current best lower bound is that it can be as large as $c^{.9039}$ for certain c (unpublished).

A lower bound on state complexity of shifts

Theorem. *Let $(a_i)_{i\geq 0}$ be an automatic sequence that is not eventually periodic. Then the shifted sequences $(a_{i+c})_{i\geq 0}$ have state complexity exceeding $c'(\log c)/(\log\log c)$ for infinitely many c.*

Proof. (For the binary case.) The number of possible n-state DFAOs with input alphabet $\{0, 1, \ldots k-1\}$ and output alphabet $\{0, 1\}$ is at most $M := n^{kn-1}2^n$.

Consider all shifts $(a_{i+c})_{i\geq 0}$ for $c < M$.

Either two shifts by different c are the same (which happens if and only if the sequence is eventually periodic), or there is some shift by $c < M$ that requires more than n states. Expressing n in terms of c now gives the bound.

The infinite Fibonacci word

The infinite Fibonacci word $\mathbf{f} = f(0)f(1)f(2)\cdots = 01001010\cdots$ is another (generalized) automatic sequence.

Its simplest definition is as the fixed point of the morphism that sends $0 \to 01$ and $1 \to 0$.

It can also be generated by an automaton, but now the inputs are in a different numeration system—not base k for any k.

The Zeckendorf numeration system

Instead we use the Zeckendorf (or Fibonacci) numeration system.

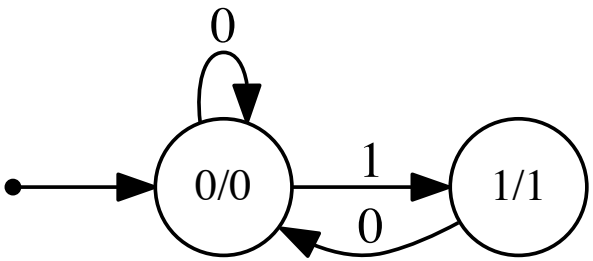
In this numeration system, instead of powers of k as in base k, we use the Fibonacci numbers $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, $F_5 = 5$, and so forth.

So n is written as a sum $\sum_{2 \leq k \leq t} e_k F_k$, or in short form, as a bit string $e_t e_{t-1} \cdots e_2$.

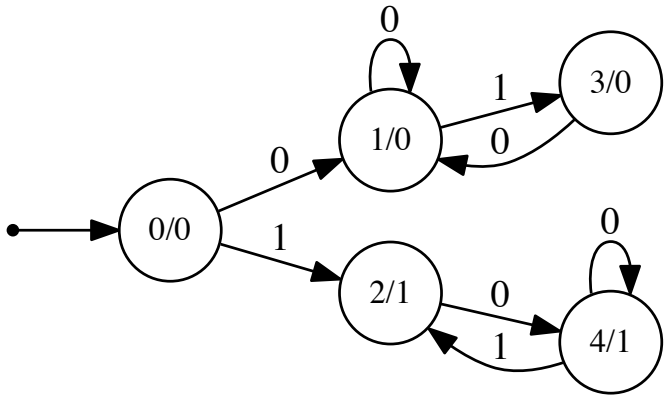
As is well-known, in this system every non-negative integer has a unique representation (ignoring leading 0's), provided one does not use two consecutive Fibonacci numbers; equivalently, the bit string cannot contain an occurrence of 11.

So 18 is legally represented as $F_7 + F_5 = 13 + 5$ but not as $F_7 + F_4 + F_3 = 13 + 3 + 2$.

The infinite Fibonacci word as an automatic sequence



Automaton generating the Fibonacci word (msd-first)



Automaton generating the Fibonacci word (lsd-first)

Our main result

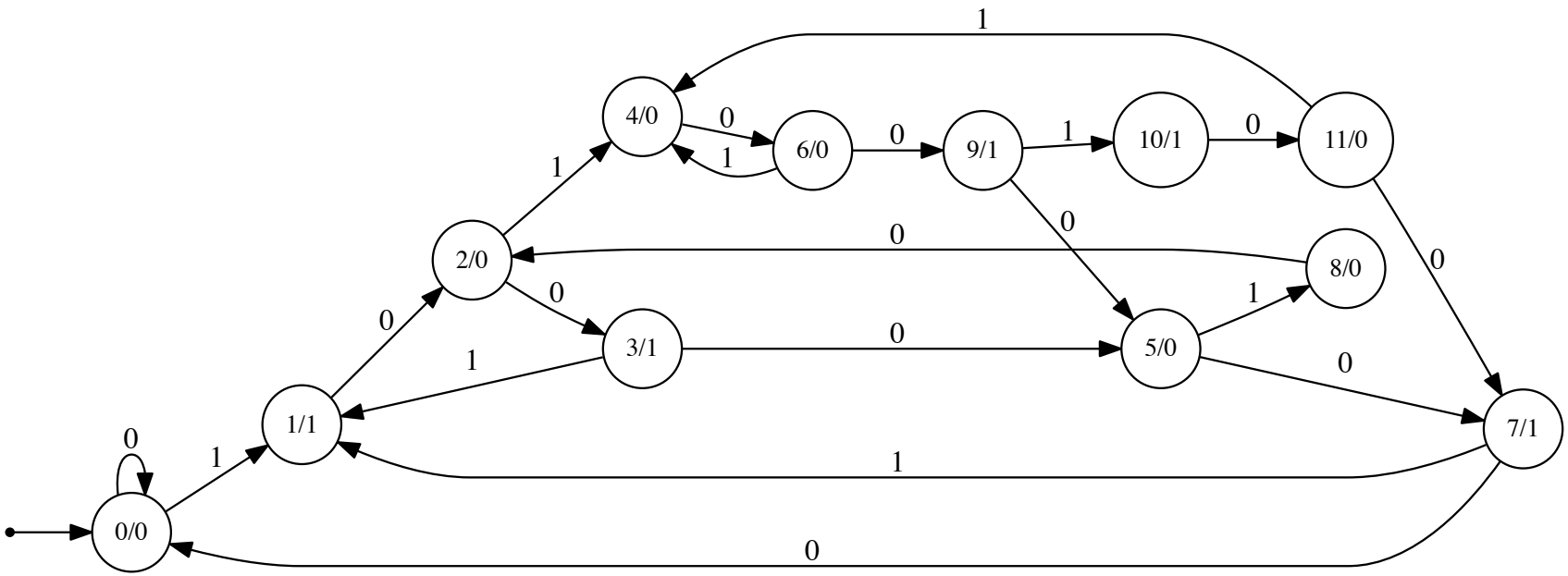
The main result of our paper is the following.

Theorem. *The state complexity of the c-shift $(f(i+c))_{i\geq 0}$ of the Fibonacci word* $\mathbf{f}$ *is $O(\log c)$, for both msd-first and lsd-first representations of the input n.*

In other words, the Fibonacci word has **very close to the minimum possible state complexity** of its shifts, for an infinite non-eventually-periodic sequence.

An example

The automaton below is for the shifted sequence $(f(i+100))_{i\geq 0}$ in msd-first format.



The Fibonacci word in terms of fractional parts

Our proof relies on an alternative characterization of the Fibonacci word.

Lemma. *Let $\varphi = (1+\sqrt{5})/2$ be the golden ratio. Then $f(i) = 1$ if and only if the fractional part $\{i\varphi\}$ belongs to the interval* $(-\varphi \bmod 1, -2\varphi \bmod 1) = (0.381966011250105 1, 0.763932022500210 2)$.

Corollary. *We have $f(i+c) = 1$ if and only if* $\{i\varphi\} \in (-(c+1)\varphi \bmod 1, -(c+2)\varphi \bmod 1)$.

These are a consequence of the fact that **f** is an example of a Sturmian word.

Sketch of the proof in the lsd-first case

$F_{k+1}/F_k \doteq \varphi$ is a famous good approximation.

It turns out that in fact $F_k\varphi$ is very close to an integer.

Therefore large F_k in the Zeckendorf representation of n are not very important for the location of $n\varphi$.

Sketch of the proof in the lsd-first case

Definition. For each m with $0 \leq m < F_{d+1}$ let $I_d(m)$ be the interval defined by

$$I_d(m) = [L_d(m)\varphi, R_d(m)\varphi) \bmod 1$$

where if d is odd we have

$$R_d(m) = -F_{d+1} + m \quad \text{and} \quad L_d(m) = \begin{cases} -F_d + m, & \text{if } m < F_d; \\ -F_{d+2} + m, & \text{if } m \geq F_d, \end{cases}$$

and if d is even, the values of $R_d(m)$ and $L_d(m)$ are switched.

Let $d \geq 2$. For every integer n, let $n^{[\leq d]}$ denote the integer obtained from n by discarding all digits with index larger than d; i.e., if $n = \sum_{2 \leq i \leq t} e_i F_i$, then $n^{[\leq d]} = \sum_{i=2}^{\min\{d,t\}} e_i F_i$.

The crucial lemma for the lsd-first case

Lemma Let $d \geq 2$. Then the intervals $I_d(m)$ for $0 \leq m < F_{d+1}$ form a partition of $[0, 1)$. Moreover, for every nonnegative integer n we have

$$\{n\varphi\} \in I_d(m) \iff n^{[\leq d]} = m.$$

Using this lemma, we can create an automaton out of the intervals $I_d(m)$ whose transitions correspond to reading the least significant bits of the representation of m. Once we have read enough bits, we end up in two of four sink states, depending on whether the output is 0 or 1.

The accept states are A_0 and A_1, and the reject states are R_0 and R_1.

Illustration for $c = 10$

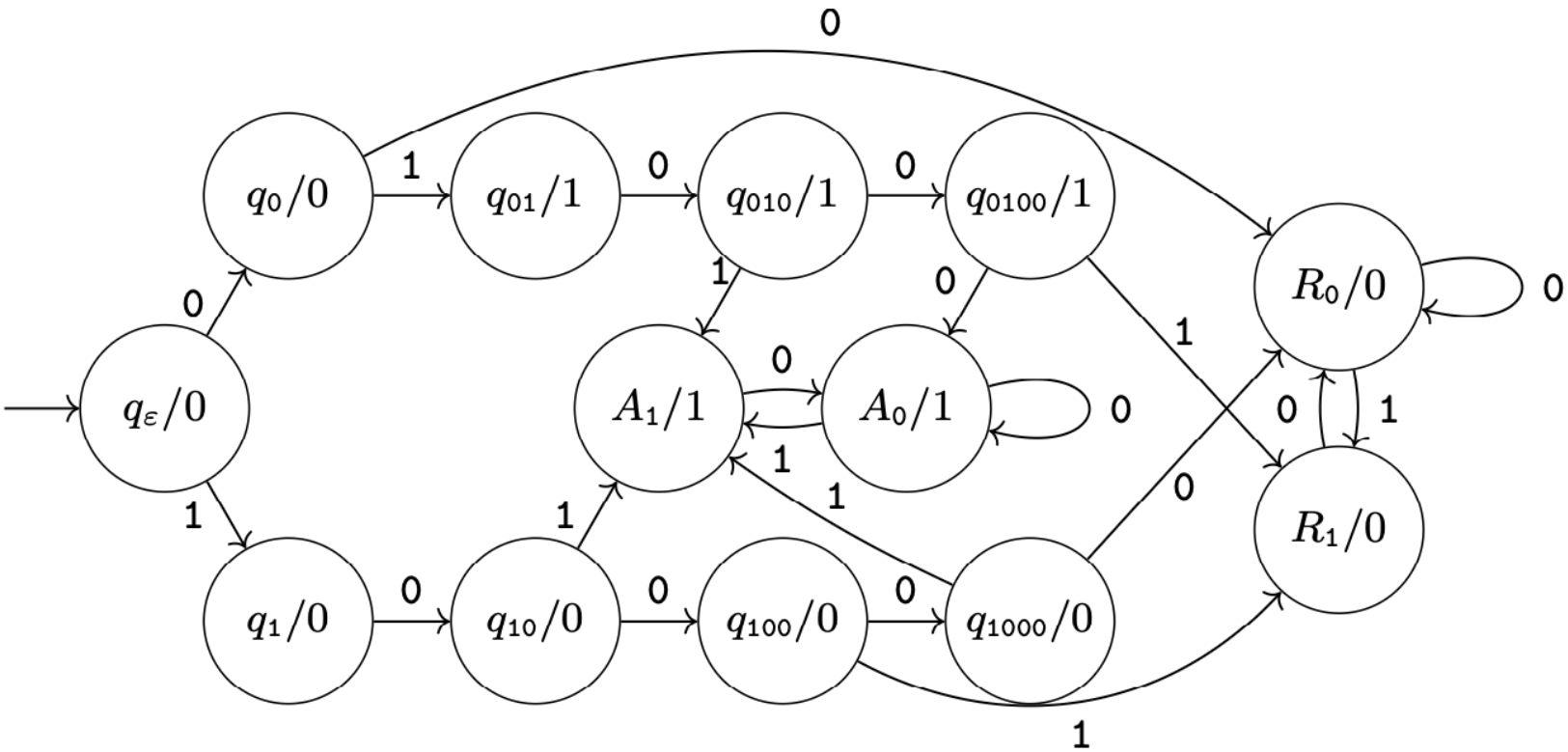


Fig. 3. Lsd-first DFAO for $(f(i+10))_{i\geq 0}$.

Exact state complexity in the lsd-first case

We can even give a formula for the *exact* state complexity of the shifted-by-c Fibonacci word in the lsd-first case.

Theorem. For $c \geq 5$, the number of states in the minimal lsd-first DFAO generating $(f(i+c))_{i\geq 0}$ is

$$2(|(c)_F| + 1) + 1 - g(c),$$

where $|(c)_F|$ means the number of bits of c in Zeckendorf representation, and $g(c) = 1$ if $f(c-1) = f(c-2) = 0$, and $g(c) = 0$ otherwise.

Why study state complexity of operations on automatic sequences?

- There is a system, called `Walnut`, that can automatically prove claims of certain assertions about automatic sequences.
- The assertions have to be phrased in first-order arithmetic with addition (an extension of Presburger arithmetic).
- The system works by compiling a first-order formula into an automaton.
- Assertions about the factors of the Fibonacci word, for example, might require evaluating a predicate such as

$$\forall t \ (t < n) \implies f(i + t) = f(j + t).$$

- Users of the system would like to know how long a proof might take.
- To estimate this, we need to know how many states the intermediate automata might have.

Intermediate automata

How is $\forall t\ (t < n) \implies g(i+t) = g(j+t)$ compiled into an automaton? Suppose g is an N-state DFAO.

An automaton for $i + t$ has a constant number of states. An automaton for $g(i + t)$ has $O(N^2)$ states.

An automaton for $g(i + t) = g(j + t)$ has $O(N^4)$ states (cross-product construction).

An automaton for $(t < n) \implies g(i + t) = g(j + t)$ has $O(N^4)$ states (cross-product construction).

An automaton for $\forall t\ (t < n) \implies g(i + t) = g(j + t)$ has $2^{O(N^4)}$ states (de Morgan + subset construction).

Intermediate automata

Open Problem.
Are there examples of automatic sequences g for which the state complexity of $\forall t\ (t < n) \implies g(i+t) = g(j+t)$ is exponential?

Yes, if alphabet can grow.

How about for fixed alphabet size?

Another open problem

An open problem is to determine the state complexity of the shifts of the Tribonacci word $\mathbf{tr} = 0102010\cdots$.

This word is defined as the fixed point of the morphism $0 \to 01$, $1 \to 02$, and $2 \to 0$.

Here we cannot use the same sort of idea as for the Fibonacci word, since the Tribonacci word is not definable in terms of the fractional parts of multiples of n by the dominant root $1.839\cdots$ of its characteristic polynomial $X^3 - X^2 - X - 1$.