Using finite automata to compute the base-b representation of the golden ratio and other quadratic irrationals

Aaron Barnoff[1], Curtis Bright[1][0000−0002−0462−625X], and Jeffrey Shallit[2][0000−0003−1197−3820]

[1] School of Computer Science, University of Windsor, Windsor, ON N9B 3P4, Canada, `{barnoffa,cbright}@uwindsor.ca`

[2] School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada, `shallit@uwaterloo.ca`

Abstract. We show that the nth digit of the base-b representation of the golden ratio is a finite-state function of the Zeckendorf representation of b^n, and hence can be computed by a finite automaton. Similar results can be proven for any quadratic irrational. We use a satisfiability (SAT) solver to prove, in some cases, that the automata we construct are minimal.

1 Introduction

The base-b digits of famous irrational numbers, where $b \geq 2$ is an integer, have been of interest for hundreds of years. For example, William Shanks computed 707 decimal digits of π in 1873 (but only the first 528 were correct) [19]. As a high school student, the third author used a computer in 1976 to determine the first 10,000 digits of the decimal representation of $\varphi = (\sqrt{5}+1)/2$, the golden ratio, using the computer language APL [18].

The celebrated results of Bailey, Borwein, and Plouffe [2] demonstrated that one can compute the nth bit of certain famous constants, such as π, in $O(n)$ time and $o(n)$ space.[3]

Can finite automata generate the base-b digits of irrational algebraic numbers, such as φ? This fundamental question was raised by Cobham in the late 1960's (a re-interpretation of a related question due to Hartmanis and Stearns [9]). Though Cobham believed for a time that he had proved they cannot be so generated [7], his proof was flawed, and it was not until 2007 that Adamczewski and Bugeaud [1] succeeded in proving that there is no deterministic finite automaton with output that, on input n expressed in base b, returns the nth base-b digit of an irrational real algebraic number α.

Even so, in this paper we show that, using finite automata, one *can* compute the nth digit in the base-b representation of the golden ratio φ! At first glance

[3] Sometimes this result is described as "computing the nth digit without having to compute the previous $n-1$ digits". But this is not really a meaningful assertion, since the phrase "computing x without computing y" is not so well-defined.

this might seem to contradict the Adamczewski–Bugeaud result. But it does not, since for our theorem the input is not n expressed in base b, but rather b^n in an entirely different numeration system, the Zeckendorf representation. As we will see below, analogous results exist for any quadratic irrational.

Our result does not give a particularly efficient way to compute the base-b digits of quadratic irrationals, but it is nevertheless somewhat surprising. Using a SAT solver, in some cases (such as for the binary digits of φ) we can prove that the automaton we construct is minimal and unique. Interestingly, in other cases (such as for the ternary digits of φ) we were able to prove the minimality of our automaton, but we discovered several distinct automata with the same number of states computing the same quadratic irrational, at least up to a high precision. It is conceivable that the automata produced by our method are indeed always minimal and unique, but we leave this as an open question.

2 Number representations and automata

A DFAO (deterministic finite automaton with output) A consists of a finite number of states along with labeled transitions connecting them. The automaton processes an input string x by starting in the distinguished start state q_0, and then following the transitions from state to state, according to each successive symbol of x. Each state q has an output $\tau(q)$ associated with it, and the function f_A computed by the DFAO maps the input x to the output associated with the last state reached. For an example of a DFAO, see Figure 2.

A DFA (deterministic finite automaton) is quite similar to a DFAO. The only difference is that there are exactly two possible outputs associated with each state, either 0 or 1. States with an output of 1 are called "accepting" or "final". If an input results in an output of 1, it is said to be accepted by the DFA. A *synchronized* DFA [6] is a particular type of DFA that takes two inputs in parallel; this is accomplished by making the input alphabet a set of pairs of alphabet symbols. A synchronized automaton computes a synchronized sequence $(f(n))_{n\geq 0}$; it does this by accepting exactly the inputs where the first components spell out a representation of n, and the second components spell out a representation for $f(n)$, where leading zeros may be required to make the inputs the same length; thus n and $f(n)$ are read in parallel. For more about synchronized sequences, see [16]. An example of a synchronized DFA appears in Figure 1. Throughout the paper, integer inputs are processed starting with the most significant digit.

Let x be a non-negative real number, let $b \geq 2$ be an integer, and write the base-b representation of x in the form $x = \sum_{-\infty<i\leq t} a_i b^i = a_t a_{t-1} \cdots a_0 . a_{-1} a_{-2} \cdots$, where $a_i \in \{0, 1, \ldots, b-1\}$. For $n \geq 0$, we call a_{-n-1} the nth digit to the right of the radix point. This choice of associating n with a_{-n-1} is perhaps a little unusual, but it seems to decrease the size of the automata produced.

2.1 Zeckendorf representation

The Fibonacci numbers are defined, as usual, by $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$. The Zeckendorf representation [11,21] of a natural number n is the unique way of writing n as a sum of Fibonacci numbers F_i, $i \geq 2$, subject to the condition that no two consecutive Fibonacci numbers are used. We may write the Zeckendorf representation as a binary string $(n)_F = a_1 \cdots a_t$, where $n = \sum_{1 \leq i \leq t} a_i F_{t+2-i}$. For example, $(43)_F = 34 + 8 + 1 = F_9 + F_6 + F_2$ has representation $\mathsf{10010001}$. The substring $\mathsf{11}$ cannot occur due to the rule that two consecutive Fibonacci numbers cannot be used. In what follows, leading zeros in strings are typically ignored without comment. We also denote the inverse of $(\cdot)_F$ by $[\cdot]_F$; i.e., $[\mathsf{10010001}]_F = 43$.

3 Automata and the base-b representation of φ

Our main result is Theorem 1 below.

Theorem 1. *For all integers $b \geq 2$, there exists a DFAO $\mathcal{A}_b$ that, on input the Zeckendorf representation of b^n, computes the nth digit to the right of the point in the base-b representation of φ.*

Proof. It is known that there exists a 7-state synchronized DFA A_1 accepting, in parallel, the Zeckendorf representations of q and $\lfloor q\varphi \rfloor$ for all $q \geq 0$ [17, Thm. 10.11.1 (a)]. Its transition diagram is depicted in Figure 1, where accepting states are denoted by double circles, and the initial state is 0, labeled by a headless arrow entering.

The DFA A_1 is constructed using the fact that $\lfloor q\varphi \rfloor = [(q-1)_F \mathsf{0}]_F + 1$, where $(q-1)_F\mathsf{0}$ is the left shift of the string $(q-1)_F$. For example, $\lfloor 11\varphi \rfloor = 17$, and we find $(10)_F = \mathsf{10010}$, left-shift that to get $\mathsf{100100} = (16)_F$, and add 1 to get 17.

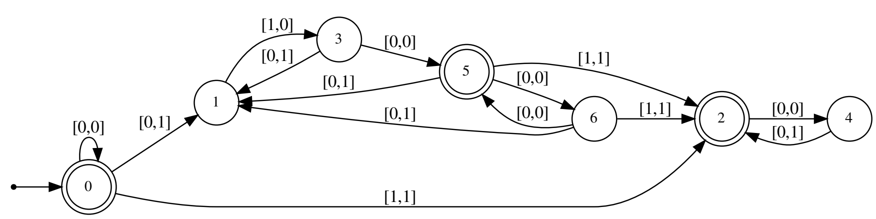

Fig. 1. Synchronized automaton for $\lfloor q\varphi \rfloor$. The inputs are the Zeckendorf representation of q and x, in parallel; it accepts iff $x = \lfloor q\varphi \rfloor$.

To understand how to use this automaton, observe that $(11)_F = \mathsf{10100}$ and $(\lfloor 11\varphi \rfloor)_F = (17)_F = \mathsf{100101}$. Since these two numbers have representations of different lengths, we need to pad the former with a leading $\mathsf{0}$. Then if $x = [\mathsf{0},\mathsf{1}][\mathsf{1},\mathsf{0}][\mathsf{0},\mathsf{0}][\mathsf{1},\mathsf{1}][\mathsf{0},\mathsf{0}][\mathsf{0},\mathsf{1}]$, the first components concatenated spell out $\mathsf{010100}$ and the second components spell out $\mathsf{100101}$. When we input this, starting at state 0 we visit, successively, states $1, 3, 5, 2, 4, 2$, and so we accept.

Let x be a positive real number, with base-b representation $y.a_0a_1a_2\cdots$, where the period (or radix point) is the analogue of the decimal point for base b, and y is an arbitrary finite block of digits. Now $b^{n+1}x$ has base-b representation $ya_0a_1\cdots a_{n-1}a_n.a_{n+1}\cdots$ and $\lfloor b^{n+1}x \rfloor$ has base-b representation

$ya_0a_1 \cdots a_{n-2}a_{n-1}a_n$. Similarly, $b\lfloor b^n x \rfloor$ has base-b representation $ya_0a_1 \cdots a_{n-1}0$. Hence $\lfloor b^{n+1}x \rfloor - b\lfloor b^n x \rfloor = a_n$. In the particular case where $x = \varphi$, we get a formula for the nth digit to the right of the radix point of φ, namely

$$D_b(n) \coloneqq \lfloor b^{n+1}\varphi \rfloor - b\lfloor b^n \varphi \rfloor.$$

From the DFA A_1 computing $\lfloor q\varphi \rfloor$, it is possible to create another DFA A_2 accepting, in parallel, the Zeckendorf representations of q and $\lfloor bq\varphi \rfloor - b\lfloor q\varphi \rfloor$. This is based on the fact that there is an algorithm to compile a first-order logic statement involving the usual logical operations (AND, OR, NOT, etc.), the integer operations of addition, subtraction, multiplication by constants, and the universal and existential quantifiers, into an automaton that accepts the Zeckendorf representation of those integers making the statement true [13].

From the DFA A_2, we can compute b individual DFAs $A_{b,i}$ accepting the Zeckendorf representation of those q for which $\lfloor bq\varphi \rfloor - b\lfloor q\varphi \rfloor = i$, for $0 \leq i < b$. Finally, we combine all the $A_{b,i}$ together into a single DFAO A_3 (using a product construction for automata) computing the difference $\lfloor bq\varphi \rfloor - b\lfloor q\varphi \rfloor$.

By substituting $q = b^n$, we see that this automaton A_3 is the desired one, computing $D_b(n)$ on input the Zeckendorf representation of b^n. □

We now use `Walnut`, which is free software for compiling first-order logical expressions into automata, to explicitly compute the automata for the representation of φ in base 2 and base 3. For base 2, we need the following `Walnut` commands (further explanation follows below):

```
reg shift {0,1} {0,1} "([0,0]|[0,1][1,1]*[1,0])*":
def phin "?msd_fib (s=0 & q=0) | Ex $shift(q-1,x) & s=x+1":
def phid2 "?msd_fib Ex,y $phin(2*q,x) & $phin(q,y) & x=2*y+1":
combine FD2 phid2:
```

These produce the DFAO in Figure 2.

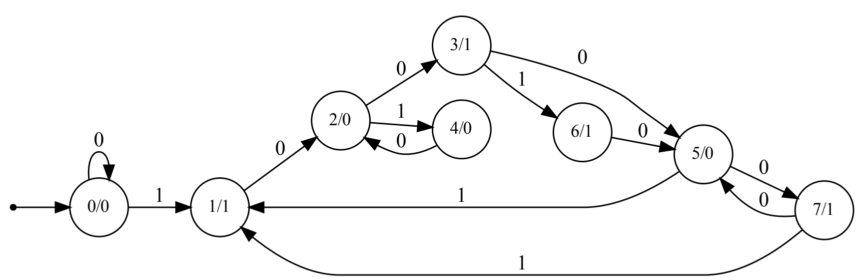

Fig. 2. Automaton $\mathcal{A}_2$ for the nth bit (base 2 digit) to the right of the binary point of φ. States are labeled in the form a/c, where a is the state number and c is the output. The input is the Zeckendorf representation of 2^n, and the output is c when the last state reached is labeled a/c.

For example, in base 2, we have $\varphi = 1.1001111000110111 \cdots$. To compute the 4th digit to the right of the binary point we write $2^4 = 16$ in Zeckendorf representation, namely 100100, and feed it into the automaton, starting at state 0 and reaching states $1, 2, 3, 6, 5, 7$ successively, with output 1 at the end.

We now explain the `Walnut` commands above that generate the DFAO in Figure 2. The first line creates a DFA called `shift`, using a regular expression; it takes two base-2 inputs and accepts only if the second is the left shift of the first. Next is the DFA `phin`, which is shown in Figure 1 and uses `shift` to check that its two inputs have the relationship $(n)_F$ and $[(n-1)_F 0]_F + 1$, which computes the function $n \to \lfloor n\varphi \rfloor$ in a synchronized fashion. Next, the DFA `phid2`, when given the representation of q as input, accepts if $\lfloor 2q\varphi \rfloor - 2\lfloor q\varphi \rfloor = 1$, and rejects otherwise. Lastly, `combine` converts `phid2` into the DFAO of Figure 2 by replacing the accepting and rejecting states of `phid2` with output values 1 and 0, respectively.

The automaton for base 3 (see Figure 3) can be constructed similarly with the following `Walnut` commands:

```
reg shift {0,1} {0,1} "([0,0]|[0,1][1,1]*[1,0])*":
def phin "?msd_fib (s=0 & n=0) | Ex $shift(n-1,x) & s=x+1":
def phid31 "?msd_fib Ex,y $phin(3*n,x) & $phin(n,y) & x=3*y+1":
def phid32 "?msd_fib Ex,y $phin(3*n,x) & $phin(n,y) & x=3*y+2":
combine FD3 phid31=1 phid32=2:
```

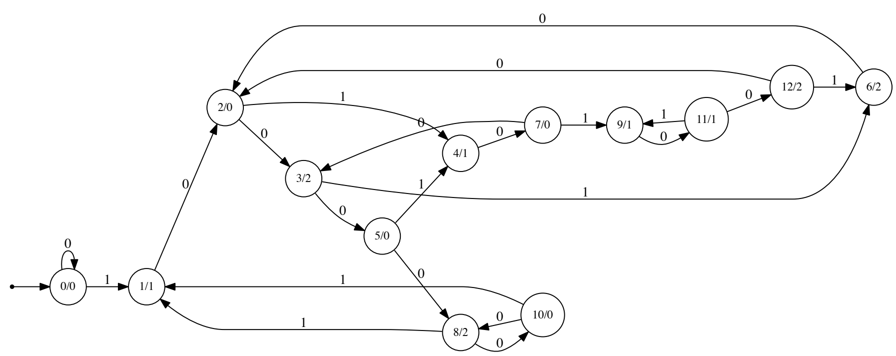

Fig. 3. Automaton for the nth digit to the right of the point of φ in base 3, with inputs as in Figure 2.

In base 3, $\varphi = 1.1212001122021210\cdots$. To compute the 3rd digit to the right of the point we write $3^3 = 27$ in Zeckendorf representation as 1001001 and pass it to the automaton in Figure 3, which, starting at state 0, traverses states $1, 2, 3, 6, 2, 3, 6$ successively, giving an output of 2.

There is no conceptual barrier to carrying out similar computations for any base $b \geq 2$. For base 10, for example, `Walnut` computes a finite automaton with 97 states that, on input $(10^n)_F$, returns the nth digit to the right of the decimal point in the decimal expansion of φ.

4 Other quadratic irrationals

There is nothing special about φ, and the same ideas can be used for any quadratic irrational. What makes quadratic irrationals special in this context is Lagrange's theorem: these numbers, and only these, have a continued fraction expansion that is ultimately periodic. This is crucial, because if this property does not hold, then the sequence of continued fraction convergents cannot satisfy a linear recurrence [12]. But a linear recurrence is needed in order to construct a numeration system with good decidability properties.

4.1 Handling $\sqrt{2}$

Another representation for the natural numbers is based on the Pell numbers, defined by $P_0 = 0$, $P_1 = 1$, and $P_n = 2P_{n-1} + P_{n-2}$ for $n \geq 2$. We can then write every natural number $n = \sum_{1 \leq i \leq t} a_i P_{t+1-i}$ where $a_i \in \{0, 1, 2\}$. To get uniqueness of the representation, we have to impose two conditions. First, we must have that $a_t \neq 2$. Second, if $a_i = 2$, then $a_{i+1} = 0$. See [4] for more details. The unique representation, over the alphabet $\{0, 1, 2\}$, is denoted $(n)_P$.

The Pell numeration system in `Walnut` can be used to construct automata computing the base-b digits of $\sqrt{2}$, just as we did for φ. This results in a 6-state DFAO for base 2 (see Figure 4), and a 14-state DFAO for base 3. The `Walnut` commands for base 2 are:

```
reg pshift {0,1,2} {0,1,2}
   "([0,0]|([0,1][1,1]*([1,0]|[1,2][2,0]))|[0,2][2,0])*":
def sqrt2n "?msd_pell (s=0 & n=0) | Ex $pshift(n-1,x) & s=x+2":
def sqrt2d2 "?msd_pell Ex,y $sqrt2n(2*n,x) & $sqrt2n(n,y)
   & x=2*y+1":
combine SD2 sqrt2d2:
```

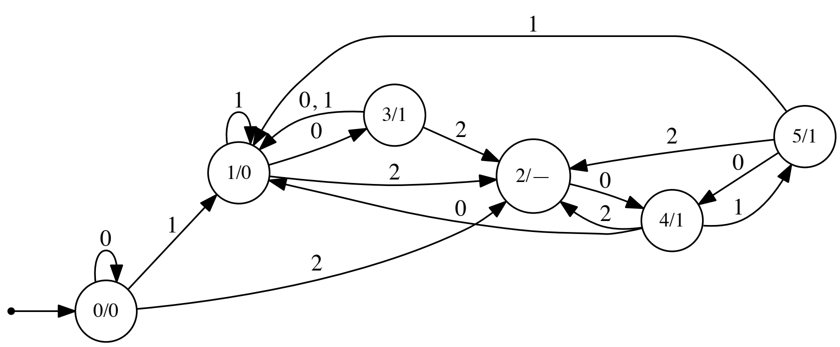

Fig. 4. Automaton for the nth bit to the right of the binary point of $\sqrt{2}$. Input is 2^n in Pell representation.

The alert reader will observe that no output is associated with state 2. This is because inputs that lead to this state, such as 12, are not valid Pell representations. However, the state cannot be removed, because 120 *is* a valid Pell representation.

4.2 Ostrowski representation

Of course, what makes our results work is that the numeration systems are "tuned" to the particular quadratic irrational we want to compute. For φ, the numeration system is based on the Fibonacci numbers; for $\sqrt{2}$, the Pell numbers. We need to find an appropriate numeration system that is similarly "tuned" to any quadratic irrational. It turns out that the proper system is the Ostrowski numeration system [3,14].

Every irrational real number α can be expressed uniquely as an infinite simple continued fraction $\alpha = [d_0, d_1, d_2, \ldots]$. Furthermore, q_n is called the nth denominator of a convergent for α if $q_{-2} = 1$, $q_{-1} = 0$, and $q_n = d_n q_{n-1} + q_{n-2}$ for $n \geq 0$. For example, the continued fraction for π is $[3, 7, 15, 1, \ldots]$, corresponding to the sequence $(q_n)_{n\geq 0} = 1, 7, 106, 113, \ldots$ (OEIS A002486).

The Ostrowski α-numeration system uses the sequence $(q_n)_{n\geq 0}$ of the denominators of the convergents for α to construct a unique representation for a non-negative integer N expressed as

$$N = [a_{n-1}a_{n-2}\cdots a_0]_\alpha = \sum_{0\leq i<n} a_i q_i,$$

where the a_i have to obey the Ostrowski rules

$$0 \leq a_0 < d_1; \tag{1}$$

$$0 \leq a_i \leq d_{i+1} \text{ for } i \geq 1; \text{ and} \tag{2}$$

$$\text{for } i \geq 1, \text{ if } a_i = d_{i+1} \text{ then } a_{i-1} = 0. \tag{3}$$

The Ostrowski α-representation for $N = [a_{n-1}a_{n-2}\cdots a_0]_\alpha$ is then determined with a greedy algorithm, starting at the most significant term and choosing the largest multiple a_{n-1} for q_{n-1} that is less than N, and then applying the same algorithm recursively to $N - a_{n-1}q_{n-1}$. For example, for $\alpha = \sqrt{3}+1 = [2, \overline{1, 2}]$, the denominators of the continued fraction convergents form the sequence $(q_n)_{n\geq 0} = 1, 1, 3, 4, 11, 15, \ldots$ (OEIS A002530). Rule 1 for the construction forces $a_0 = 0$ because $d_1 = 1$, while rule 2 requires that $a_1 \leq d_2 = 2$, $a_2 \leq d_3 = 1$, and so on. Rule 3 ensures uniqueness by enforcing the constraint that if $a_1 = d_2 = 2$, then $a_2 = 0$, and if $a_2 = d_3 = 1$, then $a_3 = 0$, and so on. Then, for example, the α-representation of 37 is $2 \cdot 15 + 4 + 3 = 2q_5 + q_3 + q_2 = [20110]_\alpha$.

In order to construct a DFAO $\mathcal{A}_b$ that, given the input of the Ostrowski α-representation of b^n, computes the nth digit to the right of the point in the base-b representation of α, we require an Ostrowski α-synchronized function $n \to \lfloor n\alpha \rfloor$. Consider a quadratic irrational $0 < \beta < 1$ with a purely periodic continued fraction $[0, \overline{d_1, d_2, \ldots, d_m}]$; here the straight bar or vinculum denotes the periodic part. Then Schaeffer et al. [15] showed that the sequence $(\lfloor n\beta \rfloor)_{n\geq 1}$ is Ostrowski β-synchronized, via the relation

$$[(n-1)_\beta 0^m]_\beta = q_m(n-1) + q_{m-1} \cdot \lfloor n\beta \rfloor, \tag{4}$$

where q_i is the denominator of the ith convergent to β, and $(n-1)_\beta 0^m$ is the β-representation of $n-1$, left-shifted m times.

Furthermore, it was shown that if $\alpha > 0$ belongs to $\mathbb{Q}(\beta)$, then $(\lfloor n\alpha \rfloor)_{n\geq 1}$ is synchronized in terms of the Ostrowski β-representation through the relation $\alpha = (a + d\beta)/c$, where d, $c \geq 1$, and

$$\lfloor n\alpha \rfloor = \left\lfloor \frac{\lfloor dn\beta \rfloor + an}{c} \right\rfloor . \tag{5}$$

This is notable because when constructing an Ostrowski α-representation with `Walnut`, it is assumed that $0 < \alpha < \frac{1}{2}$, which corresponds to a continued fraction with terms $d_0 = 0$ and $d_1 > 1$. If $\alpha \geq \frac{1}{2}$, then we can set $d_0 = 0$ and rotate the period until $d_1 > 1$, giving a quadratic irrational $0 < \beta < \frac{1}{2}$ corresponding to the periodic part of α. Then an Ostrowski representation for β can be constructed, and Eq. (4) is used to find an automaton for $\lfloor n\beta \rfloor$, followed by Eq. (5) to find an automaton for $\lfloor n\alpha \rfloor$. Therefore, $(\lfloor n\alpha \rfloor)_{n\geq 1}$ is synchronized in terms of the Ostrowski β-representation.

For example, for $\alpha = \sqrt{3} + 1 = [2, \overline{1, 2}]$, we have $\alpha \geq \frac{1}{2}$. Since we only care about the digits after the radix point, we set $d_0 = 0$ and then rotate the period to get $\beta = [0, \overline{2, 1}] = (\sqrt{3} - 1)/2 < 1/2$. This gives the sequence of denominator convergents $1, 2, 3, 8, 11, 30, \ldots$, where $m = 2$, $q_m = 3$, and $q_{m-1} = 2$, and so Eq. (4) gives $[(n-1)_\beta 00]_\beta = 3(n-1) + 2\lfloor n\beta \rfloor$. This results in a DFA for $\lfloor n\beta \rfloor$ that has 23 states. Then, we find $\alpha = (2 + 2\beta)/1$, with $a = 2$, $b = 2$, and $c = 1$, and Eq. (5) gives a DFA with 20 states, shown in Figure 5.

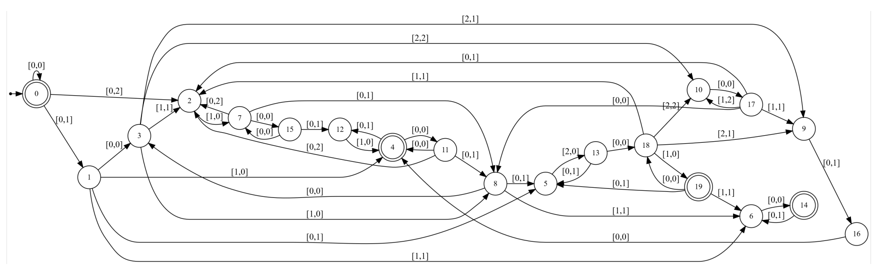

Fig. 5. Synchronized automaton for $\lfloor n\alpha \rfloor$ for $\alpha = \sqrt{3} + 1$.

Then, for example $(5)_\beta = 110$ and $(\lfloor 5\alpha \rfloor)_\beta = (13)_\beta = \texttt{10010}$. When we input $[0, 1][0, 0][1, 0][1, 1][0, 0]$ into the automaton, we visit states $1, 3, 8, 6, 14$ in succession, and so we accept. From here, the same general process that is outlined in Theorem 1 can be used to construct a DFA accepting in parallel the Ostrowski α-representations of q and $\lfloor bq\alpha \rfloor - b\lfloor q\alpha \rfloor$, and ultimately the DFAO $\mathcal{A}_b$ as desired.

4.3 `Walnut` implementation

Constructing the DFAOs for other quadratic irrationals with `Walnut` requires the `ost` command to create custom Ostrowski representations. As explained above, Walnut requires that $0 < \beta < \frac{1}{2}$ to create the corresponding Ostrowski representation, and it is possible to create a DFAO for $\alpha \geq \frac{1}{2}$ by synchronizing it in terms of the Ostrowski representation for β. Presented below are the general steps for constructing a DFAO for the digits of the base-2 representation of a quadratic irrational α with Walnut, using the process explained above with Equations (4) and (5).

First, we construct the continued fraction of $\beta < \frac{1}{2}$ from α by setting $d_0 = 0$ and rotating the period until $d_1 > 1$, if necessary. Next, we determine the denominators $j = q_m$ and $k = q_{m-1}$ of the continued fraction convergent to β, where m is the number of elements in the period. Lastly, we find a, b, and c from the relation $\alpha = (a + b\beta)/c$, where $b, c \geq 1$. With these, we can use the following Walnut commands:

```
# Construct Ostrowski representation for Beta
ost ostBeta [0] [d1 d2 ... dm];
# Create a DFA of z = floor(n*Beta) using j and k
def betan "?msd_ostBeta Eu,v n=u+1 & $shift(u,v) & v=k*z+j*u":
# Create a DFA of z = floor(n*Alpha) synchronized
def alphan "?msd_ostBeta Eu $betan(b*n,u) & z=(u+a*n)/c":
# Create a DFAO for Alpha in base 2
def alphan_d2 "?msd_ostBeta Ex,y $alphan(2*n,x) & $alphan(n,y) & x!=2*y":
combine AD2 alphan_d2:
```

The shift DFA can be constructed from a regular expression as done above for φ, and is based on the specific representation and continued fraction sequence. If multiple left-shifts are required, it may be simpler to create a shift DFA that left-shifts only one position at a time, and chain its use together multiple times. For example, three left-shifts could be achieved using a 1-shift DFA by:

```
def betan "?msd_ostBeta Eu,v,w,x n=u+1 & $shift(u,v) & $shift(v,w)
       & $shift(w,x) & x=k*z+j*u":
```

Using this process, we created the DFAOs for other quadratic irrationals including the “bronze ratio” $(\sqrt{13} + 3)/2 = [3, \overline{3}]$ and several Pisot numbers. We give the Walnut code below.

The bronze ratio $(\sqrt{13} + 3)/2$ *in bases 2 and 3:*

```
# In this case m = 1, q_m = 3, and q_(m-1) = 1.
ost bt [0] [3];
reg bts {0,1,2,3} {0,1,2,3}
   "([0,0]|[0,2][2,2]*[2,0]|([0,2][2,2]*[2,3]|[0,3])
   [3,0]|([0,1]|[0,2][2,2]*[2,1])([1,1]|[1,2][2,2]*[2,1])*
   (([1,2][2,2]*[2,3]|[1,3])[3,0]|[1,2][2,2]*[2,0]|[1,0]))*":
def btbn "?msd_bt Eu,v n=u+1 & $bts(u,v) & v=1*z+3*u":
def btan "?msd_bt Eu $btbn(1*n,u) & z=(u+3*n)/1":
```

DFAO for the bronze ratio in base 2 (7 states):

```
def btn_d2 "?msd_bt Ex,y $btan(2*n,x) & $btan(n,y) & x!=2*y":
combine BTND2 btn_d2:
```

DFAO for the bronze ratio in base 3 (8 states):

```
def btn_d31 "?msd_bt Ex,y $btan(3*n,x) & $btan(n,y) & x=3*y+1":
def btn_d32 "?msd_bt Ex,y $btan(3*n,x) & $btan(n,y) & x=3*y+2":
combine BTND3 btn_d31 btn_d32:
```

Pisot number $\sqrt{3}+1$ *and* $(\sqrt{3}-1)/2$ *in base 2:*

```
# In this case m = 2, q_m = 3, and q_(m-1) = 2.
ost pv1 [0] [2 1];
reg pv1s {0,1,2} {0,1,2} "([0,0]|([0,1][1,1][1,0]|[0,1][1,0])|
     [0,2][2,0])*":
def pv1bn "?msd_pv1 Et,u,v n=t+1 & $pv1s(t,u) & $pv1s(u,v)
   & v=2*z+3*t":
```

DFAO for $(\sqrt{3}-1)/2 = [0, \overline{2,1}]$ in base 2 (see Figure 7):

```
def pv1bn_d2 "?msd_pv1 Ex,y $pv1bn(2*n,x) & $pv1bn(n,y) & x!=2*y":
combine PV1B2 pv1bn_d2:
```

DFAO for $\sqrt{3}+1 = [2, \overline{1,2}]$ in base 2 (27 states):

```
def pv1an "?msd_pv1 Eu $pv1bn(2*n,u) & z=(u+2*n)/1":
def pv1n_d2 "?msd_pv1 Ex,y $pv1an(2*n,x) & $pv1an(n,y) & x!=2*y":
combine PV12 pv1n_d2:
```

Pisot number $(\sqrt{17}+3)/2$ *and* $(\sqrt{17}-3)/4$ *in base 2:*

```
# In this case m = 3, q_m = 7, and q_(m-1) = 4.
ost pv2 [0] [3 1 1];
reg pv2s {0,1,2,3} {0,1,2,3}
   "([0,0]|[0,1][1,0]|[0,1][1,1][1,0]|[0,2][2,0]|
   [0,2][2,1][1,0]|[0,3][3,0])*":
def pv2bn "?msd_pv2 Es,t,u,v n=s+1 & $pv2s(s,t)
   & $pv2s(t,u) & $pv2s(u,v) & v=4*z+7*s":
```

DFAO for $(\sqrt{17}-3)/4 = [0, \overline{3,1,1}]$ in base 2 (see Figure 6):

```
def pv2bn_d2 "?msd_pv2 Ex,y $pv2bn(2*n,x) & $pv2bn(n,y) & x!=2*y":
combine PV2B2 pv2bn_d2:
```

DFAO for $(\sqrt{17}+3)/2 = [3, \overline{1,1,3}]$ in base 2 (27 states):

```
def pv2an "?msd_pv2 Eu $pv2bn(2*n,u) & z=(u+3*n)/1":
def pv2n_d2 "?msd_pv2 Ex,y $pv2an(2*n,x) & $pv2an(n,y) & x!=2*y":
combine PV22 pv2n_d2:
```

5 Are the automata minimal?

The automata that `Walnut` constructs for computing $\lfloor bq\varphi \rfloor - b\lfloor q\varphi \rfloor$ on input $q \geq 0$ are guaranteed to be minimal. However, in this paper, with our application to computing the base-b digits of φ, we are only interested in running these automata in the special case when $q = b^n$, the powers of b. Could it be that there are even smaller automata that answer correctly on inputs of the form b^n (but might give a different answer for other inputs)? After all, for each t, we are only concerned with behavior of the automaton on linearly many inputs of

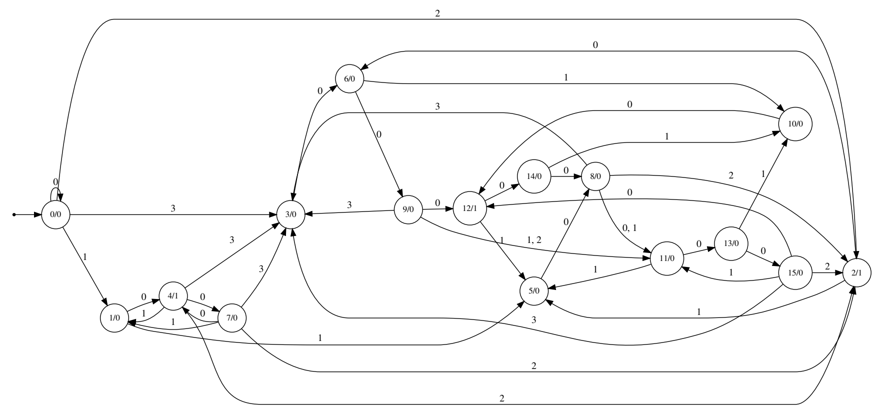

Fig. 6. Automaton for the nth bit to the right of the binary point of $(\sqrt{17}-3)/4$ in base 2. Input is n in the Ostrowski representation corresponding to the real number $[0, 3, 1, 1, 3, 1, 1, 3, 1, 1, \ldots]$.

length t, as opposed to the exponentially large set of valid length-t Zeckendorf representations. Thus, the automaton is not very constrained.

We do not know the answer to this question, in general. The question is likely difficult; in terms of computational complexity, it is a special case of a problem known to be NP-hard, namely, the problem of inferring a minimal DFAO from incomplete data [8]. However, this problem can sometimes be solved in practice using satisfiability (SAT) solving [20].

We are able to show that some of our automata are indeed minimal, among all automata giving the correct answers on inputs of the form $q = b^n$, and satisfying two conventions: first, that leading zeroes in the input cannot affect the result, and second, that the automata obey the Ostrowski rules (1)–(3) for the particular numeration system. Our method of proving minimality, and in some cases uniqueness, uses SAT solving.

We use a modified version of a MinDFA solver called `DFA-Inductor` [20] to generate SAT encodings for minimal automata, which are then passed to the `CaDiCaL` SAT solver [5] to determine whether they have a satisfying solution. `DFA-Inductor` uses the *compact encoding* method given by Heule and Verwer [10], which defines eight constraints—four mandatory and four redundant—to translate DFA identification into a graph coloring problem, and then encodes those constraints into a SAT instance.

`DFA-Inductor` only supports DFAs (and hence only accepting or rejecting states), however, and additional output status labels were added for bases larger than 2. `DFA-Inductor` does not explicitly encode a "dead state" rejecting invalid strings, but a transition to a dead state can be implied by a lack of an outgoing transition on a given state. Another redundant constraint of the compact encoding method forces each state to have an outgoing transition on every symbol, which must be amended to exclude whichever symbols must transition to the implied dead state.

Our automata follow the convention that the start state consumes leading `0`s in the input string. In terms of the compact encoding variables, $y_{\ell,p,q}$ indicates that state p has a transition to state q on label ℓ. This constraint is then implemented by enforcing state 0 to have a self-loop on the symbol `0` using the unit clause $y_{\mathtt{0},0,0}$, and the dictionary given to `DFA-Inductor` states that the string `0` produces output 0.

In order for the SAT solver to construct automata that obey the rules of a given Ostrowski representation, we encode the Ostrowski rules (2)–(3) as a set of constraints. Rule 1 is satisfied simply by only including strings in the dictionary that are valid in the given representation. Without these constraints, the solver may find a smaller DFAO by allowing rule-breaking transitions—such as allowing consecutive 1s for φ in the Zeckendorf representation.

5.1 Ostrowski encoding for purely periodic quadratic irrationals

Each Ostrowski α-representation is a language made up from the set of valid strings that can be constructed using the Ostrowski rules (1)–(3). This language is recognized by a canonical DFA, and serves as the base that informs the valid structure of the final DFAO. Constructing a DFAO using only the states in the Ostrowski base DFA guarantees that rules 2 and 3 of the Ostrowski construction are never violated. Conveniently, `Walnut` automatically generates a DFA of the Ostrowski base during the process of constructing the representation.

Since each state in the base DFA has a unique transition set, we can refer to the ith state in the base DFA as the ith *base state*. For example, Figure 7 shows for $\alpha = (\sqrt{3}-1)/2 = [0, \overline{2, 1}]$ how each base state in the Ostrowski base DFA (bottom), labelled `B0` to `B5`, correspond exactly to a state in the DFAO for returning the ith digit of α in base 2 (top).

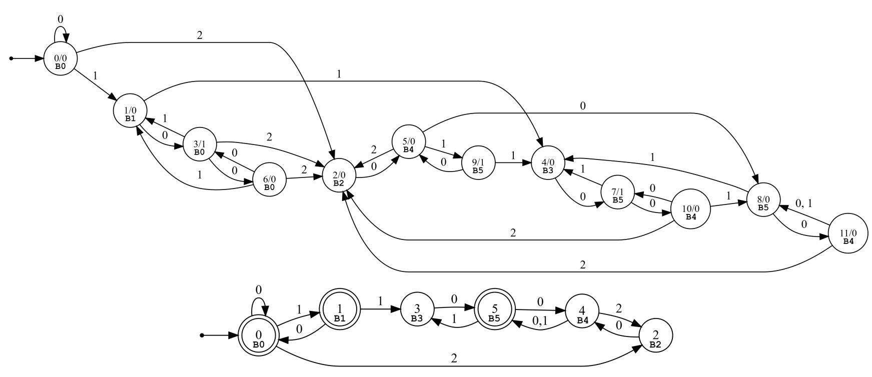

Fig. 7. Relationship between the Ostrowski base states and DFAO states for $\alpha = (\sqrt{3}-1)/2$.

The Ostrowski rules (2)–(3) are encoded through the states in the Ostrowski base DFA by constraining each state in the DFAO to match a certain base state. Therefore, to encode the base states, we create a new variable $b_{p,t}$, which says state p in the DFAO is related to base state t in the Ostrowski base DFA. We then relate the b variable to the transition variable $y_{\ell,p,q}$, which constrains the set of valid transitions between p and q according to which base states they are associated with. The encoding is presented in Table 1.

The last constraint in the table is the only one that needs to be manually determined for each Ostrowski base DFA. For example, for $\alpha = (\sqrt{3}-1)/2$ in Figure 7, base state `B4` is encoded as follows, where Q denotes the set of states in the DFAO and B denotes the set of states in the Ostrowski base DFA:

$$\bigwedge_{\substack{i,j\in Q\\ i\neq j}} \big((b_{i,4} \wedge b_{j,2} \to \neg y_{0,i,j}) \wedge (b_{i,4} \wedge b_{j,2} \to \neg y_{1,i,j}) \wedge (b_{i,4} \wedge b_{j,5} \to \neg y_{2,i,j})\big)$$

$$\bigwedge_{\substack{i,j\in Q\\ i\neq j}} \ \bigwedge_{k\in B\setminus\{\mathtt{2},\mathtt{5}\}} \ \bigwedge_{\ell\in\{\mathtt{0},\mathtt{1},\mathtt{2}\}} (b_{i,4} \wedge b_{j,k} \to \neg y_{\ell,i,j})$$

Constraints	**Range**	**Meaning**
$\neg y_{k,0,0}$	$1 \leq k \leq c$	The start state can only have a self-loop on 0.
$\neg y_{k,i,i}$	$i \in Q; i \neq 0; 1 \leq k \leq c$	No states other than the start state can have a self-loop on any label.
$b_{0,0}$		The start state is related to base state 0.
$b_{i,s} \to \neg b_{i,t}$	$i \in Q$; $s, t \in B$	Each state in the DFAO must be related to at most one base type.
$b_{i,1} \vee b_{i,2} \vee \cdots \vee b_{i,\lvert B\rvert}$	$i \in Q$	Each state in the DFAO must be related to at least one base type.
$(b_{i,s} \wedge b_{j,t}) \to \neg y_{k,i,j}$	$i, j \in Q$; $s, t \in B$; $k \in \Sigma$; $\delta(s, k) \neq t$	Suppose DFAO state i is related to base state s, and state j is related to base state t. If state s in the base DFA does not have a transition to state t on label k, then i cannot have a transition to j on label k in the DFAO.

Q = set of states in DFAO; B = set of states in Ostrowski base DFA;
δ is the transition function of the DFAO; Σ = alphabet; $c = \max(\Sigma)$

Table 1. SAT encoding of Ostrowski constraints for purely periodic quadratic irrationals.

5.2 Results

Table 2 gives our results of DFA minimization by SAT on a few quadratic irrationals. In each of the cases, the `Walnut` solution was confirmed to be minimal by proving that there are no satisfying assignments of the SAT encoding with a smaller number of states than in the `Walnut`-produced automaton.

The dictionary containing the Ostrowski representation of the first i digits is referred to as the ith digit set. The solver is run on the SAT encoding of each digit set for a given number of states. The state count was increased every time the solver returned UNSAT, and the digit set was increased every time a satisfying assignment was found. Once the state count given by the `Walnut`-produced solution was reached, the solver was run exhaustively to find

Quadratic Irrational	φ base 2 8 states	φ base 3 13 states	$\sqrt{2}$ base 2 6 states	$\frac{\sqrt{13}+3}{2}$ base 2 7 states	$\frac{\sqrt{13}+3}{2}$ base 3 8 states	$\frac{\sqrt{3}-1}{2}$ base 2 12 states	$\frac{\sqrt{17}-3}{4}$ base 2 16 states
Digit set size	54	197	29	64	64	27	57
SAT time (sec)	0.50	28,425.5	0.08	142.81	44.68	0.14	68.11
UNSAT time (sec)	0.18	12,123.0	0.02	0.52	24.76	0.08	2.59
Number of candidates	1	3	1	3	7	1	9

Table 2. Results for computing minimal automata for various quadratic irrationals.

all satisfying assignments of the SAT formula and therefore *all* candidates for the minimal automata computing the quadratic irrational. However, most satisfying assignments encoded automata that only computed the given digit set correctly and did *not* correctly compute the digits of the quadratic irrational to a higher precision than what was provided in the given digit set.

The digit set size given in Table 2 is the smallest dictionary required for the SAT solver to find the n-state `Walnut` solution. The SAT time is the time required by the solver to find the `Walnut` automaton. The UNSAT time is the time required to determine that no automata exists using $n-1$ states. Since no candidate solutions are found at $n-1$ states, we conclude that the n-state `Walnut` solution is minimal.

In some cases, we found multiple distinct candidates that correctly compute at least 250,000 digits of the quadratic irrational (see the last row of Table 2). For all except $(\sqrt{17}-3)/4$, these candidate solutions differ from the `Walnut` solution only by their outgoing transitions on the start state. The candidates for φ (base 3) and $(\sqrt{13}+3)/2$ (base 2) have differing transitions on label 1, while the candidates for $(\sqrt{13}+3)/2$ (base 3) differ on label 2. All of the candidates for $(\sqrt{17}-3)/4$ have the same start state, but differ in their transitions on label 2. Given how similar the candidate solutions are to the `Walnut` solution and that they are correct up to a high precision, it is possible that the `Walnut` solution is not unique, though we leave this as an open problem.

Minimization of the DFAOs in some other examples presented a challenge for the SAT solver. For φ in base 4, it took over 25 hours for the 78th digit set to be declared UNSAT at 13 states. For $\sqrt{2}$ in base 3, it took over 55 hours for the 258th digit set to be declared SAT at 11 states, but the satisfying assignment found by the solver corresponded to an automaton that incorrectly computed the ternary digits of $\sqrt{2}$ starting at the 321st digit.

Acknowledgments

We thank the referees for several useful suggestions.

References

1. Adamczewski, B., Bugeaud, Y.: On the complexity of algebraic numbers I. Expansions in integer bases. Ann. Math. **165**, 547–565 (2007)

2. Bailey, D., Borwein, P., Plouffe, S.: On the rapid computation of various polylogarithmic constants. Math. Comp. **66**, 903–913 (1997)
3. Baranwal, A.R., Schaeffer, L., Shallit, J.: Ostrowski-automatic sequences: theory and applications. Theoret. Comput. Sci. **858**, 122–142 (2021)
4. Baranwal, A.R., Shallit, J.: Critical exponent of infinite balanced words via the Pell number system. In: Mercaş, R., Reidenbach, D. (eds.) WORDS 2019, Lecture Notes in Computer Science, vol. 11682, pp. 80–92. Springer-Verlag (2019)
5. Biere, A., Fazekas, K., Fleury, M., Heisinger, M.: CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In: Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions. Department of Computer Science Report Series B, vol. B-2020-1, pp. 51–53. University of Helsinki (2020)
6. Carpi, A., Maggi, C.: On synchronized sequences and their separators. RAIRO Inform. Théor. App. **35**, 513–524 (2001)
7. Cobham, A.: On the Hartmanis-Stearns problem for a class of tag machines. In: IEEE Conference Record of 1968 Ninth Annual Symposium on Switching and Automata Theory. pp. 51–60 (1968), also appeared as IBM Research Technical Report RC-2178, August 23 1968
8. Gold, M.E.: Complexity of automaton identification from given data. Inform. Control **37**, 302–320 (1978)
9. Hartmanis, J., Stearns, R.E.: On the computational complexity of algorithms. Trans. Amer. Math. Soc. **117**, 285–306 (1965)
10. Heule, M., Verwer, S.: Exact DFA identification using SAT solvers. In: Sempere, J.M., García, P. (eds.) ICGI 2010, Lecture Notes in Artificial Intelligence, vol. 6339, pp. 66–79. Springer-Verlag (2010)
11. Lekkerkerker, C.G.: Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci. Simon Stevin **29**, 190–195 (1952)
12. Lenstra, Jr., H.W., Shallit, J.O.: Continued fractions and linear recurrences. Math. Comp. **61**, 351–354 (1993)
13. Mousavi, H., Schaeffer, L., Shallit, J.: Decision algorithms for Fibonacci-automatic words, I: Basic results. RAIRO Inform. Théor. App. **50**, 39–66 (2016)
14. Ostrowski, A.: Bemerkungen zur Theorie der Diophantischen Approximationen. Abh. Math. Sem. Hamburg **1**, 77–98, 250–251 (1922), reprinted in *Collected Mathematical Papers*, Vol. 3, pp. 57–80
15. Schaeffer, L., Shallit, J., Zorcic, S.: Beatty sequences for a quadratic irrational: decidability and applications (2024), arxiv preprint arXiv:2402.08331 [math.NT].
16. Shallit, J.: Synchronized sequences. In: Lecroq, T., Puzynina, S. (eds.) WORDS 2021, Lecture Notes in Computer Science, vol. 12847, pp. 1–19. Springer-Verlag (2021)
17. Shallit, J.: The Logical Approach To Automatic Sequences: Exploring Combinatorics on Words with `Walnut`, London Math. Society Lecture Note Series, vol. 482. Cambridge University Press (2023)
18. Shallit, J.: Calculation of $\sqrt{5}$ and ϕ (the golden ratio) to 10,000 decimal places (1976), reviewed in *Math. Comp.* **30** (1976), 377
19. Shanks, W.: On the extension of the numerical value of π. Proc. Roy. Soc. London **21**, 318–319 (1873)
20. Zakirzyanov, I., Shalyto, A., Ulyantsev, V.: Finding all minimum-size DFA consistent with given examples: SAT-based approach. In: Cerone, A., Roveri, M. (eds.) Software Engineering and Formal Methods: SEFM 2017 Collocated Workshops, Lecture Notes in Computer Science, vol. 10729, pp. 117–131. Springer-Verlag (2018)
21. Zeckendorf, E.: Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. Bull. Soc. Roy. Liège **41**, 179–182 (1972)