Highlights

Computing the base-b representation of quadratic irrationals using automata

Aaron Barnoff, Curtis Bright, Jeffrey Shallit

- We construct finite automata that compute the base-b digits of the golden ratio and other quadratic irrationals.
- SAT solving proves some constructed automata are minimal and unique.
- We give a heuristic argument that some of our constructed automata are not minimal, answering a question we left open in our original paper.

Computing the base-b representation of quadratic irrationals using automata

Aaron Barnoff[a], Curtis Bright[a,1,*], Jeffrey Shallit[b,2]

[a] *University of Windsor, 401 Sunset Ave, Windsor, N9B 3P4, ON, Canada*
[b] *University of Waterloo, 200 University Ave W, Waterloo, N2L 3G1, ON, Canada*

Abstract

We show that the nth digit of the base-b representation of any quadratic irrational α is a finite-state function of the Ostrowski α-representation of b^n, and hence can be computed by a finite automaton. We use a satisfiability (SAT) solver to prove, for some quadratic irrationals, that the automata we construct are both minimal and unique. For other quadratic irrationals, the SAT solver is able to find smaller automata computing the digits of the irrational up to a high precision. We conjecture in these cases that the automata found do indeed compute all digits of the irrational correctly. We give a heuristic argument for this conjecture, though we leave this as an open question.

Keywords: finite automata, golden ratio, Zeckendorf representation, Ostrowski numeration, SAT solving, minimal DFA

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, although only the first 528 were correct [1]. As a high school student, the third author used a computer in

*Corresponding author

Email addresses: barnoffa@uwindsor.ca (Aaron Barnoff), cbright@uwindsor.ca (Curtis Bright), shallit@uwaterloo.ca (Jeffrey Shallit)

1ORCID: 0000-0002-0462-625X

2ORCID: 0000-0003-1197-3820

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 [2].

The celebrated results of Bailey, Borwein, and Plouffe [3] demonstrated that one can compute the nth bit of certain famous constants, such as π, in $O(n \log n\, \mathsf{M}(\log n))$ time and $O(\log n)$ space, where $\mathsf{M}(j)$ is the cost of multiplying j-bit numbers.[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 [4]). Though Cobham believed for a time that he had proved they cannot be so generated [5], his proof was flawed, and it was not until 2007 that Adamczewski and Bugeaud [6] 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 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 α, except with the Zeckendorf representation replaced by the Ostrowski numeration system associated with α.

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 are minimal and unique. Interestingly, in other cases, the SAT solver discovered several distinct automata, sometimes with fewer states, computing the same quadratic irrational up to a high precision. We were not able to rigorously prove that these candidate automata compute the same quadratic irrational to arbitrary high precision, but we give a probabilistic heuristic argument that they do. Since the automata that we construct are in some cases larger than the candidates found by the SAT solver, we suspect that in general our construction does not produce minimal

[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.

automata.

This paper is an extended version of our previous work [7] appearing in the 2024 International Conference on Implementation and Application of Automata. In our original paper, we left open the question of if the automata we constructed were minimal. In this updated version, we answer this question under a plausible assumption about the distribution of digits in the input strings provided to the automata.

Additionally, we correct an oversight in the SAT encoding we used in our previous work, which forbid states with self-loops (except for self-loops on the start state to consume leading zeros). After updating our encoding, this changed our previous results in a single case, namely, when computing the binary digits of $(\sqrt{3}-1)/2$. Our construction produces a 12-state automaton computing these digits. This automaton was unique and minimal under our previous restriction on self-loops, but removing this restriction permits the SAT solver to find an automaton with only 11 states computing the digits of $(\sqrt{3}-1)/2$ to a high precision. We study this automaton in Section 6.3, and we provide an argument that it in fact very likely computes the digits of $(\sqrt{3}-1)/2$ to arbitrarily high precision.

In this updated version, we also provide candidate automata for four other quadratic irrationals that have fewer states than the automata from our construction, but which likely still compute the same digits to arbitrarily high precision (see Section 6.4). In addition, we provide a more convenient way of constructing the `shift` DFAs for these quadratic irrationals (see Section 4.3) and point out an alternate way in which automata could be considered to compute the digits of quadratic irrationals (see Section 5).

2. Number representations and automata

A DFAO (deterministic finite automaton with output) A consists of a finite number of states along with labelled 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 [8] 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 [9]. 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 [10, 11] 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 $n = \sum_{i=1}^{t} a_i F_{t+2-i}$. For example, $(43)_F = 34 + 8 + 1 = F_9 + F_6 + F_2$ has representation 10010001. The substring 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., $[10010001]_F = 43$.

3. Automata and the base-b representation of φ

Our main result is Theorem 1 below.

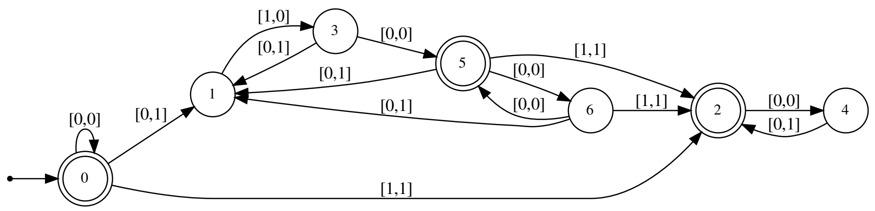

Figure 1: Synchronized automaton A_1 for $\lfloor q\varphi \rfloor$. The inputs are the Zeckendorf representation of q and x, in parallel; it accepts if and only if $x = \lfloor q\varphi \rfloor$.

Theorem 1. *For all integers $b \geq 2$, there exists a DFAO 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 $q \geq 0$ [12, 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, labelled by a headless arrow entering.

The DFA A_1 is constructed using the fact that $\lfloor q\varphi \rfloor = [(q-1)_F 0]_F + 1$, where $(q-1)_F 0$ is the left shift of the string $(q-1)_F$. For example, $\lfloor 11\varphi \rfloor = 17$, and we find $(11-1)_F = (10)_F = 10010$, left-shift that to get $100100 = (16)_F$, and add 1 to get $[100100]_F + 1 = 17$. To understand how to use this automaton, observe that $(11)_F = 10100$ and $(\lfloor 11\varphi \rfloor)_F = (17)_F = 100101$. Since these two numbers have representations of different lengths, we need to pad the former with a leading 0. Then if $x = [0,1][1,0][0,0][1,1][0,0][0,1]$, the first components concatenated spell out 010100, and the second components spell out 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 $y.a_0 a_1 a_2 \cdots$, 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 $ya_0 a_1 \cdots a_{n-1} a_n . a_{n+1} \cdots$ and $\lfloor b^{n+1} x \rfloor$ has base-b representation $ya_0 a_1 \cdots a_{n-1} a_n$. Similarly, $b\lfloor b^n x \rfloor$ has base-b representation $ya_0 a_1 \cdots a_{n-1} 0$. 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.

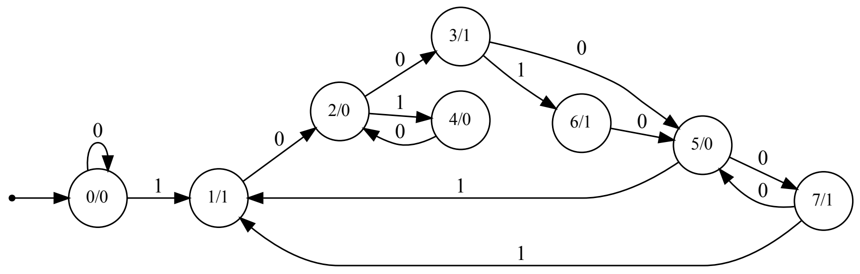

Figure 2: Automaton A_2 for the nth bit (base 2 digit) to the right of the binary point of φ. States are labelled in the form a/c, where a is the state number and c is the output. The input is the Zeckendorf representation of 2^n, and the output is c when the last state reached is labelled a/c.

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 $[(n-1)_F0]_F + 1$, 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:
```

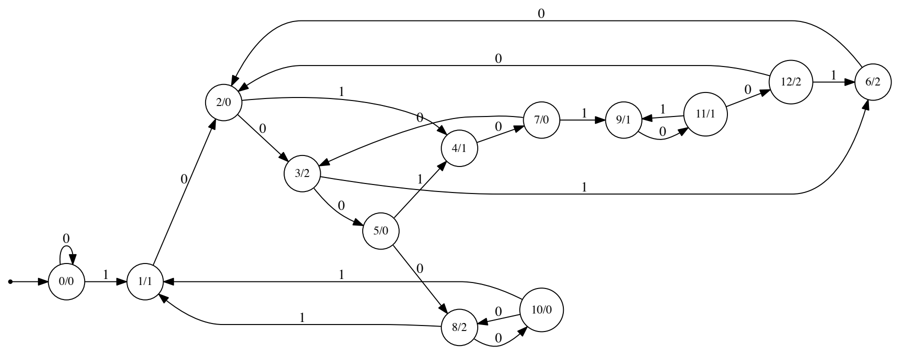

Figure 3: Automaton for the nth digit to the right of the point of φ in base 3, with inputs as in Figure 2.

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 [14]. 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 $n = \sum_{i=1}^{t} a_i P_{t+1-i}$ where $a_i \in \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}$. To get uniqueness of the representation, we have to impose two conditions. First, we must have that $a_t \neq \mathtt{2}$. Second, if $a_i = \mathtt{2}$, then $a_{i+1} = \mathtt{0}$; see [15] for more details. We denote the unique Pell representation of n by $(n)_P \in \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}^*$.

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 as follows:

```
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:
```

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.

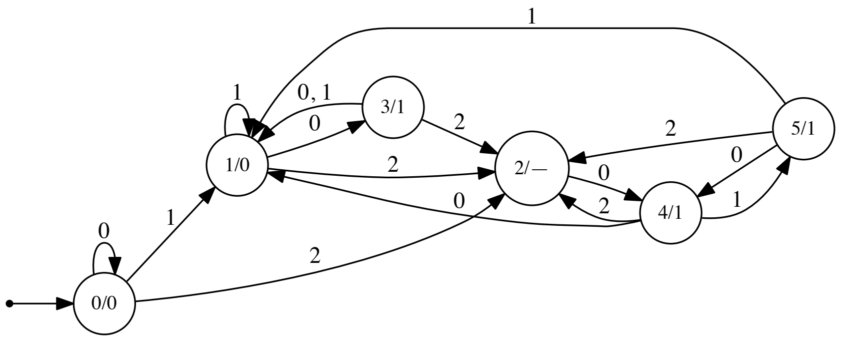

Figure 4: Automaton for the nth bit to the right of the binary point of $\sqrt{2}$. Its input is 2^n in the Pell representation, i.e., $(2^n)_P$.

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. In general, the proper system is the Ostrowski numeration system [16, 17].

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

$$a_0 \in \{0, 1, \ldots, d_1 - 1\}; \tag{1}$$

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

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

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 $a_0 = \mathtt{0}$ 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 $a_0 = \mathtt{0}$, and if $a_2 = d_3 = 1$, then $a_1 = \mathtt{0}$, and so on. Then, for example, the α-representation of 37 is $2 \cdot 15 + 4 + 3 = 2q_5 + q_3 + q_2 = [\mathtt{20110}]_\alpha$.

In order to construct a DFAO 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. [18] showed that the sequence $(\lfloor n\beta \rfloor)_{n\geq 1}$ is Ostrowski β-synchronized via the relation

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

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

Furthermore, Schaeffer et al. showed 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

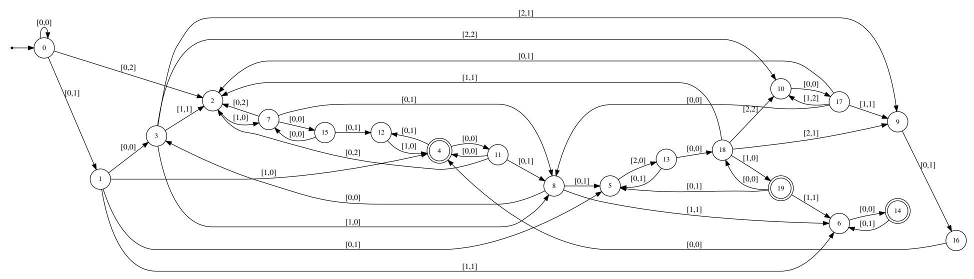

Figure 5: Synchronized automaton for $\lfloor n\alpha \rfloor$ for $\alpha = \sqrt{3}+1$.

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.

Then for example $(5)_\beta = 110$ and $(\lfloor 5\alpha \rfloor)_\beta = (13)_\beta = 10010$. 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 A_b as desired.

Finally, there are practical limitations on the types of quadratic irrationals to which we can apply this method. The key issue is that one of the intermediate automata—namely, the one responsible for shifting by the length of the period—grows exponentially. In particular, if we shift by k positions over an alphabet of size t, then this automaton will have at least t^k states. Consequently, in practice, we are restricted to quadratic irrationals with relatively small partial quotients and relatively short periods. We do not know how to prove a lower bound on the sizes of the final automata that will result from our construction.

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":
```

One further simplification for the DFAOs constructed in this paper is to use the `shift` DFA for the generalized golden mean corresponding to the largest term in the quadratic irrational's continued fraction. Because the regular expression for the `shift` DFA grows unwieldy for larger golden means, it is easier to encode the states and transitions directly using `Walnut`'s DFA format. Below is a template for constructing the `shift` DFA for the golden mean $[k, \overline{k}]$:

```
{0, 1, ..., k} {0, 1, ..., k}
0 1
0 j -> j  // insert transitions for j=0 to k
m 0       // insert states for m=1 to k-1
m j -> j  // insert transitions for j=0 to k
k 0
k 0 -> 0
```

Using the processes outlined above, we created the DFAOs for other quadratic irrationals, including the "bronze ratio" $(\sqrt{13}+3)/2 = [3, \overline{3}]$ and several Pisot numbers. The `Walnut` code is given in Appendix A.

5. An alternative construction

In Section 3, we combined multiple instances of automaton A_1 to build a more complex automaton simulating the computation of the nth digit in a given base b. In this section, we describe a different way in which the digits of φ can be computed by an automaton, also based on the automaton A_1 that in parallel takes q and x in Zeckendorf representation and accepts if and only if $x = \lfloor q\varphi \rfloor$ (see Figure 1).

We can compute the first $k+1$ digits of φ in a base b using only a single instance of A_1. This is done by intersecting A_1 with another automaton whose first component spells out b^k in Zeckendorf representation and whose second component is arbitrary. The resulting product automaton accepts exactly one string that encodes the Zeckendorf representation of the first $k+1$ digits concatenated together.

For example, to compute the first 6 digits of $\varphi = 1.100111100011\cdots$ in base 2, we construct a product automaton from A_1 and the automaton whose first component spells out $(2^5)_F$. With `Walnut`, we use the commands:

```
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 x5 "?msd_fib $phin(x,y) & x=32":
```

This produces the automaton shown in Figure 6, where the second component spells out $[10100101]_F = 51$ which, when 51 is represented in binary, gives 110011, the first six base-2 digits of φ.

This approach requires additional intersections and computations to identify the unique accepting path, unlike the method in Section 3, which provides

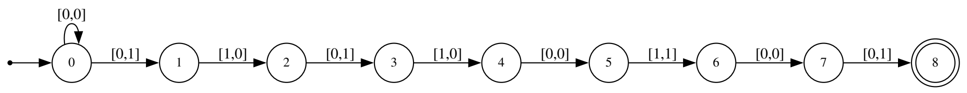

Figure 6: Automaton encoding the first 6 digits of φ in base 2. The first component spells out $(2^5)_F = \mathsf{1010100}$, and the second spells out $(51)_F = \mathsf{10100101}$.

it directly. Moreover, the resulting DFA has a size proportional to the length of the Zeckendorf representation of the concatenation of the first $k + 1$ digits in the specified base, which can grow arbitrarily large.

6. 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 behaviour of the automaton on linearly many inputs of length t, as opposed to the exponentially large set of valid length-t Zeckendorf representations. Thus, the automaton is not very constrained.

In general, we do not know the answer to this question, but in Section 6.3 we give a heuristic argument that some of the automata constructed by `Walnut` are not minimal. 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 [19]. However, this problem can sometimes be solved in practice using satisfiability (SAT) solving [20].

We are able to show that some of `Walnut`'s 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 satisfiability (SAT) solving. SAT solvers are automated reasoning tools that take as input a logical formula in conjunctive normal form and output a truth assignment under which the formula is true, if one exists. If no such truth assignment exists, the formula is said to be unsatisfiable and the solver returns UNSAT. Even though there are no provably efficient algorithms to solve the SAT problem, in practice certain kinds of problems can be effectively solved using SAT solvers [21].

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 [22] to determine whether they have a satisfying solution. `DFA-Inductor` uses the *compact encoding* method given by Heule and Verwer [23], which defines eight constraints—four mandatory and four redundant—to translate DFA identification into a graph colouring problem, and then encodes those constraints into a SAT instance.

`DFA-Inductor` only natively supports DFAs without output (i.e., only accepting or rejecting states), however, we added output status labels 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. Since one of the redundant constraints of the compact encoding method forces each state to have an outgoing transition on every symbol, this constraint 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 zeros 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 $y_{0,0,0}$, and the dictionary given to `DFA-Inductor` states that the string `0` produces output 0.

In order for the SAT solver to construct automata obeying the rules of a given Ostrowski representation, we encode the Ostrowski rules (2) and (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 `1`s for φ in the Zeckendorf representation.

6.1. 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

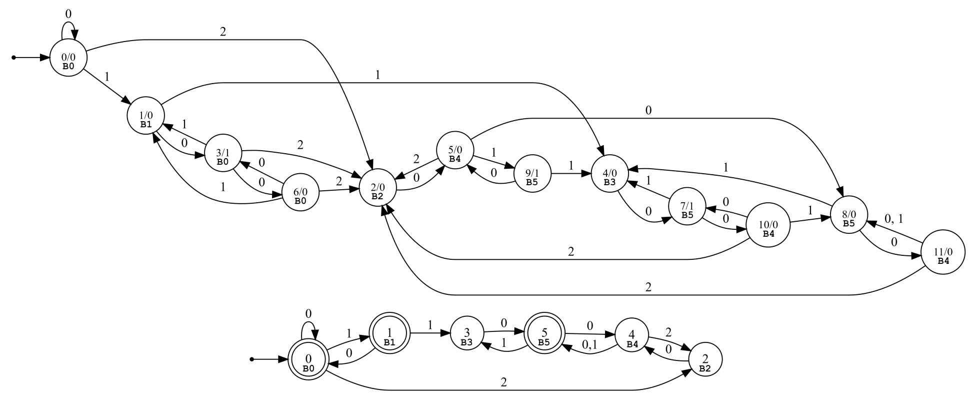

Figure 7: Relationship between the Ostrowski base states and DFAO states for $\alpha = (\sqrt{3} - 1)/2$.

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 Ostrowski base DFA has a unique transition set, we 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 `Walnut` DFAO for returning the ith digit of α in base 2 (top).

The Ostrowski rules (2) and (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 quadratic irrational, since it is derived directly from the transition function of the 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

Constraints	**Range**	**Meaning**
$b_{0,0}$		The start state is related to base state 0.
$b_{i,s} \to \neg b_{i,t}$	$i \in Q$; $s, t \in B$; $s \neq t$	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 DFAO state j is related to base state t. If s does not transition to t on label k in the base DFA, then i cannot 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.

set of states in the Ostrowski base DFA:

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

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

6.2. Results

The results were determined on a desktop computer with an Intel Core i5 10600k CPU and 32 GiB RAM.[4] Table 2 and Table 3 give our results of DFA minimization by SAT on a few quadratic irrationals. Table 2 shows the quadratic irrationals proven to have a minimal `Walnut` solution. The digit set size is the smallest dictionary required for the SAT solver to find the n-state `Walnut` solution. Table 2 also includes the time required for the solver to find the n-state `Walnut` automaton, the time required to determine that no automaton exists using $n-1$ states, the time required to exhaustively find all candidate automata with n states, and the number of candidate

[4]Our code is publicly available at https://github.com/aaronbarnoff/tcs_digits.

Quadratic irrational	φ	φ	$\sqrt{2}$	$\frac{\sqrt{13}+3}{2}$	$\frac{\sqrt{13}+3}{2}$	$\frac{\sqrt{17}-3}{4}$
Continued fraction	$[1,\overline{1}]$	$[1,\overline{1}]$	$[1,\overline{2}]$	$[3,\overline{3}]$	$[3,\overline{3}]$	$[0,\overline{3,1,1}]$
Base	2	3	2	2	3	2
`Walnut` DFAO size	8	13	6	7	8	16
Digit set size	54	197	29	64	64	57
SAT time (sec)	0.25	22169.0	0.033	30.41	2786.29	75.25
UNSAT time (sec)	0.12	3217.74	0.01	0.19	7.07	2.41
Exhaustive time (sec)	0.37	38999.16	0.05	74.55	5750.86	890.04
Number of candidates	1	3	1	3	7	9

Table 2: Quadratic irrationals whose `Walnut` DFAOs were determined to be minimal. The SAT time is the amount of time it took the SAT solver to find the `Walnut` solution. The UNSAT time is the amount of time it took the SAT solver to show there are no smaller solutions. The exhaustive time is the amount of time it took the SAT solver to find all candidate automata of minimal size.

automata found. Note that candidate automata are only guaranteed to correctly compute the digits of the specific quadratic irrational up to a precision of $p = 100{,}000$ digits. Minimality is proven by determining that there are no candidate automata with a fewer number of states than in the `Walnut`-produced automaton. We used `CaDiCaL` to determine the UNSAT time and a version of `CaDiCaL` that was modified to find *all* solutions of a SAT instance to determine the SAT time and the exhaustive time.

The process of determining the minimality of the `Walnut` DFAO begins by creating a dictionary that contains the Ostrowski representation of the first i digits of the quadratic irrational (where $i \geq 1$ will increase as the solving continues), along with each digit's output value. A SAT encoding is created from that dictionary set, and the solver attempts find a satisfying solution for a given number of states. The dictionary set size i is increased every time a satisfying assignment is found, and continues until UNSAT is returned, indicating no DFAO exists that correctly computes the first i digits with the current number of states. The state count is increased every time the solver returns UNSAT, and this process continues until the state count given by the `Walnut`-produced solution is reached.

To determine uniqueness of the `Walnut` automata, we run the solver exhaustively at the `Walnut` DFAO's state-count to find all satisfying assignments of the SAT formula and therefore *all* candidates for the minimal automata computing the quadratic irrational. Each solution is validated against a dictionary set consisting of the Ostrowski representations of the first $p = 100{,}000$ digits. Any solution whose output does not match the correct initial p digits of α is discarded, and we suspect the candidate automata remaining after this process do in fact compute the digits of α to arbitrarily high precision (though we cannot prove this). Each candidate solution is confirmed to be a unique automaton (up to isomorphism) using `nauty`, a software for computing graph isomorphism [24].

For example, in Table 2, we see that for φ in base 2, the minimum dictionary size for an 8-state solution is 54, and although the SAT solver found many 8-state DFAOs correctly computing the first 54 digits, only the `Walnut` DFAO was correct for computing the first p digits. However, for φ in base 3, out of the many DFAOs that correctly compute the first 197 digits, the solver found two other candidate solutions correctly computing the first p digits.

In other cases, we found that the SAT solver stalled in attempting to produce an UNSAT result for a state-count smaller than that of the `Walnut` solution. When these plateaus were reached, we performed an exhaustive search to identify all possible solutions at the given state-count to determine whether any solutions compute the first p digits of the quadratic irrational correctly. Table 3 contains results for quadratic irrationals with candidate automata having fewer states than the `Walnut` DFAO. In this table, the digit set size indicates the size of the smallest dictionary required by the SAT solver to find a DFAO capable of correctly computing the first p digits. As described below, the candidate DFAO for $(\sqrt{21} - 6)/3$ was constructed manually, without the use of a SAT solver.

Minimization of the DFAOs in some other examples also presented a challenge for the SAT solver, as both the size of the digit set required to find a candidate solution and the length of the input string for each digit position can become large. For φ in base 4, which has a `Walnut` DFAO with 22 states, it took over 40 hours for the 79th digit set to be declared UNSAT at 14 states. For $\sqrt{2}$ in base 3, which has a `Walnut` DFAO with 14 states, it took over 30 hours for the 259th to 267th digit sets to be declared SAT at 11 states. However, the satisfying assignments found by the solver corresponded to automata that could only correctly compute the ternary digits of $\sqrt{2}$ up to the 320th digit at best.

Quadratic irrational	$\frac{\sqrt{3}-1}{2}$	$\frac{\sqrt{2}-1}{2}$	$\frac{\sqrt{15}-3}{6}$	$\frac{\sqrt{6}-2}{4}$	$\frac{\sqrt{21}-3}{6}$
Continued fraction	$[0,\overline{2,1}]$	$[0,\overline{4,1}]$	$[0,\overline{6,1}]$	$[0,\overline{8,1}]$	$[0,\overline{3,1}]$
Base	2	2	2	2	2
Walnut DFAO size	12	12	12	12	18
Candidate DFAO size	11	11	10	11	16
Digit set size	27	31	57	588	–
Exhaustive search (sec)	0.42	62.86	159.79	1919.39	–
Number of candidates	1	10	12	≥ 435	≥ 1

Table 3: Quadratic irrationals for which we found candidate automata smaller than the `Walnut` DFAOs. The DFAO for $(\sqrt{21}-3)/6$ was constructed manually.

6.3. A heuristic argument that some `Walnut` *automata are not minimal*

We now give a heuristic argument suggesting that in general `Walnut`'s automata are not minimal amongst DFAOs required only to produce the nth digit of a specific quadratic irrational on input b^n in the relevant Ostrowski representation. We focus on $\alpha = (\sqrt{3}-1)/2 = [0,\overline{2,1}]$ in base 2, as we expect the 11-state automaton found by the SAT solver correctly computes all binary digits of α, beating the 12-state automaton computed by `Walnut`.

First, when we run the `Walnut` DFAO for α (Figure 7) on a dictionary containing the first $p = 100{,}000$ strings of the form $(2^i)_\alpha$, we find that state $6/\mathtt{0}$ is never reached as a final state. This indicates that there is no power of two up to 2^p with an α-representation in $\{\mathtt{0},\mathtt{1}\}^*$ ending with the substring `00`. In fact, upon inspecting the α-representations of 2^k for $0 \leq k \leq p$, we find that 2^{29} is the last power of 2 whose α-representation lies entirely in $\{\mathtt{0},\mathtt{1}\}^*$. In other words, the computation of the 29th digit of α is the last digit we could find that involves no transitions using the label `2`.

Next, upon examining the 11-state candidate solution found by the SAT solver (Figure 8), we see that state $6/\mathtt{0}$ in the `Walnut` DFAO has been removed, and that state $3/\mathtt{1}$ has acquired a self-loop on `0`. Since states $6/\mathtt{0}$ and $3/\mathtt{1}$ in the `Walnut` DFAO only differ in their output values and transition to each other on `0`, it follows that adding a self-loop on `0` to state $3/\mathtt{1}$ preserves the same transition structure for any strings that were previously non-final in state $6/\mathtt{0}$. Therefore, if no input $(2^n)_\alpha$ is ever final for state $6/\mathtt{0}$, we can conclude that the 11-state candidate solution computes the digits of α to arbitrarily large precision. Note that analyzing the transition structure of the

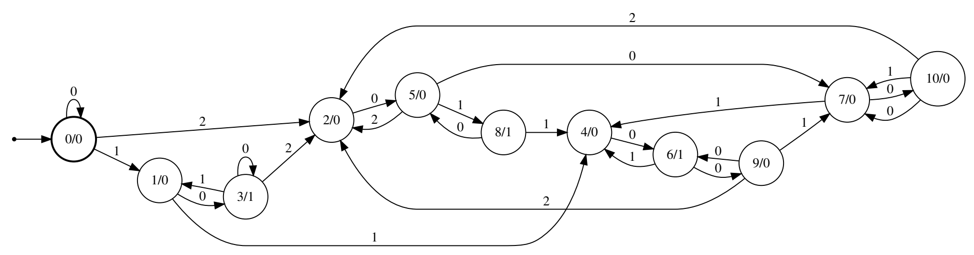

Figure 8: An 11-state SAT solution for $\alpha = (\sqrt{3}-1)/2 = [0, \overline{2, 1}]$ which correctly computes the digits of α for at least the first $p = 100{,}000$ digits.

`Walnut` automaton shows the only way the state $6/0$ can be reached is if the input string has no 2, i.e., is entirely in $\{0, 1\}^*$.

We now provide a heuristic probabilistic argument for the correctness of the candidate solution by noting that, under some plausible assumptions about the digits of the α-representations of powers of two, it is likely that there are only finitely many integers n for which $(2^n)_\alpha$ contains no 2.

Conjecture 6.3.1. *For $\alpha = (\sqrt{3}-1)/2 = [0, \overline{2, 1}]$, after $(2^{29})_\alpha =$ 110001001 1000011010101000100001, every Ostrowski α-representation of 2^n contains at least one* 2.

The digits in a given α-representation are governed by the Ostrowski construction rules, which in turn depend on the period of the continued fraction expansion of α. In particular, no representation can end in 2, representations of even length cannot begin with 2, and every 2 must be followed by a 0. As a result, these α-representations contain, on average, significantly fewer 2s than would be expected under a random distribution of digits.

Let $|(n)_\alpha|$ denote the length of the α-representation of n. Since these representations are defined recursively, we can derive closed-form formulas for the total number of representations of a fixed length k as well as for those that omit a particular digit. In particular, if we define $\mathcal{L}_k = \{\, (n)_\alpha : |(n)_\alpha| = k \,\}$, then one may show that

$$|\mathcal{L}_k| = \begin{cases} \left\lceil \frac{(2+\sqrt{3})^{(k/2)}}{3+\sqrt{3}} \right\rceil, & \text{if } k \text{ is even;} \\ \left\lfloor \frac{(2+\sqrt{3})^{(k+1)/2}}{1+\sqrt{3}} \right\rfloor, & \text{if } k \text{ is odd.} \end{cases}$$

The closed-form formulas for the even and odd cases are derived from the OEIS sequences A079935, and A001834, respectively, and come from the sequence of the denominators of the continued fraction convergents for α (OEIS A002530). In either case, we have $|\mathcal{L}_k| \geq (2+\sqrt{3})^{k/2}/5$.

Moreover, we define $\mathcal{L}_k^* \subseteq \mathcal{L}_k$ as the set of all α-representations of length k that do not contain a 2. The Ostrowski construction implies that any valid representation in $\mathcal{L}_k^*$ must end either in 00, 01, or 10. This gives a recursive formulation $\mathcal{L}_k^* = \bigcup_{x \in \mathcal{L}_{k-2}^*} \{x\mathtt{00}, x\mathtt{01}, x\mathtt{10}\}$ for $k \geq 3$, and so $|\mathcal{L}_k^*| = 3|\mathcal{L}_{k-2}^*|$. Since even-length α-representations cannot begin with 2, all strings in $\mathcal{L}_{2k}^*$ begin with 1, so $\mathcal{L}_{2k}^* = \bigcup_{x \in \mathcal{L}_{2k-1}^*} \{\mathtt{1}x\}$ for $k \geq 1$ and $|\mathcal{L}_{2k}^*| = |\mathcal{L}_{2k-1}^*|$. Combining these observations gives $|\mathcal{L}_k^*| = 3^{\lfloor k/2 \rfloor} \leq 3^{k/2}$, so $|\mathcal{L}_k^*|/|\mathcal{L}_k| \leq 5(3/(2+\sqrt{3}))^{k/2}$.

Per the Ostrowski construction, if $|(2^i)_\alpha| = k$, then q_k is the first denominator in the sequence $(q_n)_{n\geq 0}$ that is greater or equal to 2^i, where $q_k = d_k q_{k-1} + q_{k-2}$. Since $(q_n)_{n\geq 0}$ is strictly monotonically increasing and $d_k \in \{1, 2\}$, we have

$$\frac{q_k}{q_{k-1}} = d_k + \frac{q_{k-2}}{q_{k-1}} \leq 2 + \frac{q_{k-2}}{q_{k-1}} < 3.$$

Since $q_{k-1} < 2^i \leq q_k$, we have

$$q_k = \frac{q_k}{q_{k-1}} \cdot q_{k-1} < \frac{q_k}{q_{k-1}} \cdot 2^i < 3 \cdot 2^i < 2^{i+2}.$$

Thus, q_k is not greater or equal to 2^{i+2}, so $|(2^{i+2})_\alpha| > |(2^i)_\alpha|$, and at most two consecutive powers of 2 can have α-representations of the same length.

Finally, we have confirmed that all $(2^i)_\alpha$ for $30 \leq i \leq 100{,}000$ contain the digit 2, and $|(2^{100000})_\alpha| = 105{,}265$. Under the assumption that strings of the form $(2^i)_\alpha$ of length k are in $\mathcal{L}_k^*$ with probability $|\mathcal{L}_k^*|/|\mathcal{L}_k|$, one would expect the number of strings of the form $(2^i)_\alpha$ without a 2 and with $i > 100{,}000$ to be no more than approximately

$$\sum_{k=105265}^{\infty} \frac{2|\mathcal{L}_k^*|}{|\mathcal{L}_k|} \leq \sum_{k=105265}^{\infty} 10 \left(\frac{3}{2+\sqrt{3}} \right)^{k/2} < 10^{-4989}.$$

Although we believe it is very unlikely there is a power of two larger than 2^{29} whose α-representation does not contain a 2, we have assumed that α-representations of powers of two behave like "typical" α-representations in their distribution of digits. Rigorously proving this remains challenging, somewhat similar to another open question of whether every power of 16 beyond $16^4 = 65536$ must contain at least one digit from $\{1, 2, 4, 8\}$ [25].

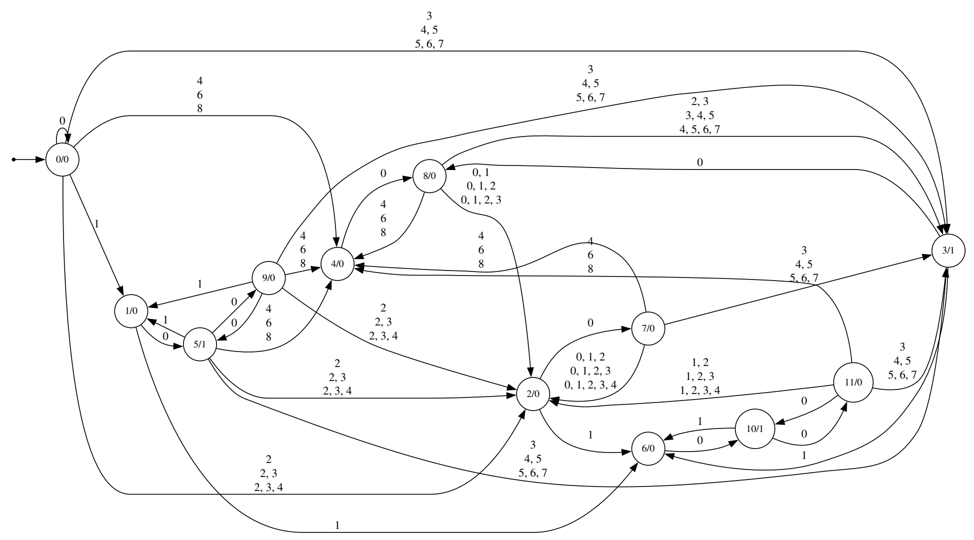

Figure 9: Automaton for the nth bit to the right of the binary point of $[0, \overline{k, 1}]$, for $k = \{4, 6, 8\}$. Stacked transition labels correspond to $k = 4$ (top), $k = 6$ (middle), $k = 8$ (bottom). Input is n in the Ostrowski representation corresponding to the real number $[0, k, 1, k, 1, k, 1, \ldots]$.

6.4. Candidates for computing continued fractions of the form $[0, \overline{m, 1}]$

The similar minimality results for the other quadratic irrationals listed in Table 3 are unsurprising given that the DFAOs computing quadratic irrationals with continued fraction $[0, \overline{m, 1}]$, where m is even, have essentially the same Ostrowski base DFA. For example, Figure 9 shows the `Walnut` DFAOs for $[0, \overline{4, 1}]$, $[0, \overline{6, 1}]$, and $[0, \overline{8, 1}]$ have the same general structure.

Consequently, the `Walnut` automata for these quadratic irrationals all contain a pair of states (5/`1` and 9/`0` in Figure 9) structurally analogous to states 3/`1` and 6/`0` in the `Walnut` automaton for $\alpha = (\sqrt{3} - 1)/2 = [0, \overline{2, 1}]$. Therefore, if no Ostrowski representation of an early power of 2 is final for either of these two states, we expect the `Walnut` solution will not be minimal for the same reasoning that we provided for α.

This expectation aligns with our observations: for example, the `Walnut` DFAO for $[0, \overline{6, 1}]$ was never final for states 5/`1` and 9/`0` while computing the first p digits, prompting the SAT solver to identify 12 unique 10-state candidate solutions with both states removed. Similarly, for $[0, \overline{8, 1}]$, the `Walnut` DFAO was never final for state 9/`0` while computing the first p digits,

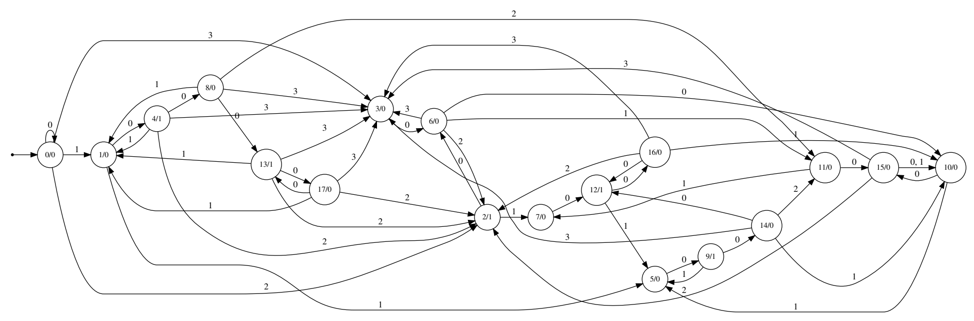

Figure 10: Automaton for the nth bit to the right of the binary point of $(\sqrt{21}-3)/6 = [0, \overline{3, 1}]$. Input is n in the Ostrowski representation corresponding to the real number $[0, 3, 1, 3, 1, 3, 1, \ldots]$.

leading the SAT solver to find 52,488 unique 11-state DFAOs with state 9/0 removed. For computational reasons, only the first 435 DFAOs were tested against the first p digits, and all passed successfully. The remaining DFAOs were verified against the first 10,000 digits and found correct up to that point. The fact that all 52,488 DFAOs correctly computed at least the first 10,000 digits—and that there are at least 435 valid candidate solutions—is somewhat unusual. However, this could be explained in part by the significantly larger size of the digit set used to construct these DFAOs compared to the other cases.

Unfortunately, the `Walnut` automaton for $\alpha = (\sqrt{21}-3)/6 = [0; \overline{3, 1}]$ has 18 states, of which only 13 were reached by the SAT solver before progress plateaued. None of the candidate SAT solutions correctly computed the first p digits. However, as shown in Figure 10, states 13/0 and 17/1 exhibit the same structural properties as the two key states in $[0, \overline{2, 1}]$, and neither of these states are final while computing the first p digits. Using this information, we manually constructed a 16-state DFAO by removing states 13/0 and 17/1, and modifying the transition structure so that 8/0 transitions back to 4/1 on 0. The resulting DFAO correctly computes the first p digits, but since the SAT solver was not used, we cannot readily determine whether additional candidate solutions exist or whether a configuration with fewer states is possible.

7. Conclusion

We provide an explicit construction for finite automata computing the nth base-b digit of a quadratic irrational (subject to the input n being given by the digits of a particular Ostrowski representation of b^n). This paper extends our previous work [7] and answers (assuming Conjecture 6.3.1) a question previously left open: are the automata we construct minimal among all automata required to give the correct answer when provided input strings of the form b^n in the relevant Ostrowski representation? Under conjecture 6.3.1, we find the answer to this question is no—our constructed automaton for computing the binary digits of $(\sqrt{3}-1)/2$ contains 12 states (see Figure 7), but an exhaustive search with a SAT solver found an 11 state automaton (see Figure 8) that we conjecture also computes the binary digits of $(\sqrt{3}-1)/2$. We provide a heuristic argument for this conjecture, but we expect that proving it rigorously will be difficult.

We also note several other quadratic irrationals and bases (such as for the binary digits of $(\sqrt{2}-1)/2$) where we expect our construction does not produce minimal automata. On the other hand, in other cases—such as in the cases of computing the binary and ternary digits of the golden ratio and the binary digits of $\sqrt{2}$—we are able to prove our constructed automata is minimal using a SAT solver and a reasonable amount of computing resources.

Acknowledgements

We thank the referees of our CIAA 2024 submission for several useful suggestions.

References

[1] W. Shanks, On the extension of the numerical value of π, Proc. Roy. Soc. London 21 (1873) 318–319.

[2] J. Shallit, Calculation of $\sqrt{5}$ and ϕ (the golden ratio) to 10,000 decimal places, reviewed in *Math. Comp.* **30** (1976), 377 (1976).

[3] D. Bailey, P. Borwein, S. Plouffe, On the rapid computation of various polylogarithmic constants, Math. Comp. 66 (1997) 903–913.

[4] J. Hartmanis, R. E. Stearns, On the computational complexity of algorithms, Trans. Amer. Math. Soc. 117 (1965) 285–306.

[5] A. Cobham, 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, 1968, pp. 51–60, also appeared as IBM Research Technical Report RC-2178, August 23 1968.

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

[7] A. Barnoff, C. Bright, J. Shallit, Using finite automata to compute the base-b representation of the golden ratio and other quadratic irrationals, in: S. Z. Fazekas (Ed.), Implementation and Application of Automata, Springer Nature Switzerland, Cham, 2024, pp. 35–50.

[8] A. Carpi, C. Maggi, On synchronized sequences and their separators, RAIRO Inform. Théor. App. 35 (2001) 513–524.

[9] J. Shallit, Synchronized sequences, in: T. Lecroq, S. Puzynina (Eds.), WORDS 2021, Vol. 12847 of Lecture Notes in Computer Science, Springer-Verlag, 2021, pp. 1–19.

[10] C. G. Lekkerkerker, Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci, Simon Stevin 29 (1952) 190–195.

[11] E. Zeckendorf, Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas, Bull. Soc. Roy. Liège 41 (1972) 179–182.

[12] J. Shallit, The Logical Approach To Automatic Sequences: Exploring Combinatorics on Words with `Walnut`, Vol. 482 of London Math. Society Lecture Note Series, Cambridge University Press, 2023.

[13] H. Mousavi, L. Schaeffer, J. Shallit, Decision algorithms for Fibonacci-automatic words, I: basic results, RAIRO Inform. Théor. App. 50 (2016) 39–66.

[14] H. W. Lenstra, Jr., J. O. Shallit, Continued fractions and linear recurrences, Math. Comp. 61 (1993) 351–354.

[15] A. R. Baranwal, J. Shallit, Critical exponent of infinite balanced words via the Pell number system, in: R. Mercaş, D. Reidenbach (Eds.), WORDS 2019, Vol. 11682 of Lecture Notes in Computer Science, Springer-Verlag, 2019, pp. 80–92.

[16] A. R. Baranwal, L. Schaeffer, J. Shallit, Ostrowski-automatic sequences: Theory and applications, Theoret. Comput. Sci. 858 (2021) 122–142.

[17] A. Ostrowski, Bemerkungen zur theorie der Diophantischen approximationen, Abh. Math. Sem. Hamburg 1 (1922) 77–98, 250–251, reprinted in *Collected Mathematical Papers*, Vol. 3, pp. 57–80.

[18] L. Schaeffer, J. Shallit, S. Zorcic, Beatty sequences for a quadratic irrational: Decidability and applications, arXiv:2402.08331 [math.NT]. (2024).

[19] M. E. Gold, Complexity of automaton identification from given data, Inform. Control 37 (1978) 302–320.

[20] I. Zakirzyanov, A. Shalyto, V. Ulyantsev, Finding all minimum-size DFA consistent with given examples: SAT-based approach, in: A. Cerone, M. Roveri (Eds.), Software Engineering and Formal Methods: SEFM 2017 Collocated Workshops, Vol. 10729 of Lecture Notes in Computer Science, Springer-Verlag, 2018, pp. 117–131.

[21] C. Bright, J. Gerhard, I. Kotsireas, V. Ganesh, Effective problem solving using SAT solvers, in: Maple in Mathematics Education and Research, Springer International Publishing, 2020, p. 205–219.

[22] A. Biere, K. Fazekas, M. Fleury, M. Heisinger, CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020, in: Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions, Vol. B-2020-1 of Department of Computer Science Report Series B, University of Helsinki, 2020, pp. 51–53.

[23] M. Heule, S. Verwer, Exact DFA identification using SAT solvers, in: J. M. Sempere, P. García (Eds.), ICGI 2010, Vol. 6339 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2010, pp. 66–79.

[24] B. D. McKay, A. Piperno, Practical graph isomorphism, II, Journal of Symbolic Computation 60 (2014) 94–112.

[25] J. Shallit, Minimal primes, Journal of Recreational Mathematics 30 (2) (2000) 113–117.

Appendix A. Walnut code for quadratic irrationals

Appendix A.1. The bronze ratio $(\sqrt{13}+3)/2 = [3, \overline{3}]$ *in bases 2 and 3*

```
# 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:
```

Appendix A.2. Pisot number $\sqrt{3}+1 = [2, \overline{1, 2}]$ *and* $(\sqrt{3}-1)/2 = [0, \overline{2, 1}]$ *in base 2*

```
# 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 $(\sqrt{3}-1)/2$ in base 2 (12 states, 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 $\sqrt{3}+1$ 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:
```

Appendix A.3. Pisot number $(\sqrt{17}+3)/2 = [3, \overline{1, 1, 3}]$ *and* $(\sqrt{17}-3)/4 = [0, \overline{3, 1, 1}]$ *in base 2*

```
# 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 $(\sqrt{17}-3)/4$ in base 2 (16 states):

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

DFAO for $(\sqrt{17}+3)/2$ 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:
```

Appendix A.4. Quadratic irrationals with continued fraction $[0, \overline{k, 1}]$ *in base 2*

```
# DFAO has 12 states if k is even, and 18 if k is odd.
# m = 2, q_m = k+1, and q_(m-1) = k.
# shiftk is the shift DFA for the golden mean [k, k...]
ost t0k1 [0] [k 1];
def t0k1bn "?msd_t0k1 Es,t,u n=s+1 & $shiftk(s,t) & $shiftk(t,u)
    & u=k*z+(k+1)*s":
def t0k1bn_d2 "?msd_t0k1 Ex,y $t0k1bn(2*n,x) & $t0k1bn(n,y)
                                             & x!=2*y":
combine T0k1B2 t0k1bn_d2:
```