Automatic proofs in combinatorial game theory

Bastien Mignoty* , Antoine Renard[1], Michel Rigo†[1], and Markus A. Whiteland‡[2]

[1]Department of Mathematics, University of Liège, Belgium
[2]Department of Computer Science, Loughborough University, Epinal Way, Loughborough, Leicestershire, LE11 3TU, United Kingdom,
{m.rigo,antoine.renard}@uliege.be, m.a.whiteland@lboro.ac.uk

Keywords: Combinatorial game theory ; Wythoff's game ; Automatic theorem proving ; Numeration systems ; First-oder logic

Mathematics Subject Classification: 91A46 ; 91A05 ; 68V15 ; 68Q45 ; 11A67

Abstract

The Büchi–Bruyère theorem asserts that the first-order theory of certain extensions of Presburger arithmetic is decidable. The software `Walnut` implements the corresponding transformation of first-order formulas into automata. This tool has already been widely and successfully used in combinatorics on words to automatically reprove results from the literature as well as proving new results. We present a new range of applications of the tool in combinatorial game theory. This is the first time this tool and such a formalism have been applied in the context of games as far as we know.

We consider Wythoff's game and many variations studied by Fraenkel and others. In this paper, we show how to use `Walnut` to obtain short automatic proofs of several results from the literature. We also prove a conjecture stated by Duchêne et al. regarding additional moves not changing the set of the $\mathcal{P}$-positions of Wythoff's game. We further state some new conjectures related to redundant moves. This work is linked with non-standard numeration systems for which addition is recognizable by a finite automaton.

1 Introduction

`Walnut` is a free software[1] system originally created by Hamoon Mousavi [27, 33]. It is extensively used for proving results in combinatorics on words and additive number theory; it has been used in a variety of papers and books, the dedicated website lists almost a hundred such entries. Some recent applications also relate to combinatorial pattern matching [32]. In this paper, we consider new applications in combinatorial game theory.

`Walnut` relies on Büchi's theorem [6, 9, 29] where the central idea is to transform first-order logical formulas into finite automata for which decision procedures can be applied. If a problem of interest can be expressed in a convenient extension of Presburger arithmetic $\langle \mathbb{N}, + \rangle$, it can then receive an automatic treatment. `Walnut` also includes automatic sequences and more general families of sequences for which the theory remains decidable [1].

As an introductory example, consider the chicken McNuggets problem whose solution can be expressed in Presburger arithmetic. Suppose that these nuggets can be purchased at McDonald's

*Former ENS Lyon student performing Master's internship at ULiège.
†Supported by FNRS Research grant T.196.23 (PDR).
‡Part of this research was performed while affiliated with ULiège and supported by FNRS Research grant 1.B.466.21F.
[1]freely available at `https://cs.uwaterloo.ca/~shallit/walnut.html`

only in quantities of 6, 9 or 20 pieces. The largest number (called *Frobenius number*) of nuggets that cannot be purchased is known to be 43. This can be expressed by the logical sentence

$$(\forall n)(n > 43 \rightarrow (\exists x, y, z \geq 0)(n = 6x + 9y + 20z)) \wedge \neg((\exists x, y, z \geq 0)(43 = 6x + 9y + 20z))$$

stating that every integer n larger than 43 is a linear combination with non-negative integer coefficients of 6, 9 and 20 and that 43 cannot be expressed in that way. This formula can be written almost verbatim in `Walnut`

```
eval nuggets "?msd_2 An (n>43)=>
  ((Ex,y,z (n=6*x+9*y+20*z)) & ~(Ex,y,z (43=6*x+9*y+20*z)))":
```

and evaluates to `TRUE`. The above example is only making use of addition, first-order quantifiers and logical connectors. The prefix `?msd_2` indicates that numbers are represented in binary with most significant digit first. In this example, the choice of the base has no importance. For references and manuals on `Walnut`, see [27, 33].

Up to our knowledge, this kind of reasoning making use of automated proofs has never been applied to the context of combinatorial game theory (with the exception of [14, 15, 34] where the considered combinatorics on words conjectures are related to games and also [35] in connections with [17]). The main idea is that if, for a given game, one has a candidate for the set of $\mathcal{P}$-positions (i.e., losing positions) whose expression in a suitable numeration system is recognized by a finite automaton, then it is possible to automatically test this conjecture. Once the result is proved, one can take advantage of this formalism to obtain new results or information. In many of Fraenkel's works [16, 17] and other related works, the study of combinatorial games is carried out by considering an adapted numeration system. It turns out that for the vast majority of the studied games, these systems are in fact usable in Walnut. They are *addable* systems as the numeration language is regular and addition is recognized by finite automata.

In this paper, we first consider Wythoff's game as a toy example. For more about Willem Wythoff, see [13]. We use `Walnut` to automatically prove Fraenkel's syntactical characterization of the $\mathcal{P}$-positions using the Fibonacci numeration system. Then we extend the rule-set in a maximal way ensuring that the set of $\mathcal{P}$-positions remains the same. In Section 2.1 thanks to `Walnut`, we are able to solve a 14-year old conjecture stated in [11] about the morphic structure of the set of forbidden moves (i.e., moves that, if adjoined, would change the set of $\mathcal{P}$-positions) and we re-obtain an intricate characterization of these moves. In Section 2.2, we reconsider the existence of redundant moves also studied in [11]. There is an automatic way to prove that there is no such move. In Sections 2.3 and 2.4, we look at two variant games: a restriction and an extension of Wythoff's game that can easily be expressed in first-order logic and thus reprove in an automatic manner results given by Ho [22].

We then review variations of Wythoff's game obtained by relaxing the set of rules [16, 17]. These games are associated with families of increasingly general numeration systems for which Fraenkel's syntactical characterization of the $\mathcal{P}$-positions still holds. The Fibonacci system is a special instance of an Ostrowski numeration system based on the convergents of the continued fraction $[\overline{1}]$. In [16] a variation of Wythoff's game based on the continued fraction $[1, \overline{m}]$ is studied. In Section 3 we are able to automatically derive the corresponding characterization for a fixed parameter m. In particular, we discover the existence of infinitely many redundant moves and are able to formulate a conjecture about them. In Section 4, we reconsider another variation of Wythoff's game [17] based on a quadratic Pisot numeration system. We explain how to build an automaton recognizing addition in this system and we can thus apply the same machinery. In terms of the languages of the numeration system that is considered, we get the following picture in Fig. 1 summarizing the increasing generalizations considered in the first three sections. In the last section, we define and analyze some new game related this time to the binary system but having a set of $\mathcal{P}$-positions satisfying the same type of syntactical properties. Surprisingly, the corresponding set of $\mathcal{P}$-positions already appeared in [18].

This paper demonstrates how powerful the `Walnut`/Büchi–Bruyère approach is. We are able to obtain automatically results from at least five different papers [11, 16, 17, 22, 36] and parts

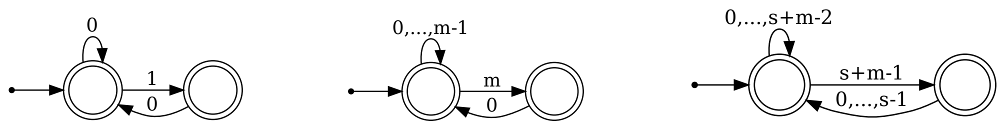

Figure 1: Automata for Fibonacci, for Ostrowski base on $[1,\overline{m}]$ and for some quadratic Pisot numbers (from left to right).

of [18]; we avoid long and tedious case analyses. Moreover, in Section 2.1 we are able to solve a long-standing conjecture stated by Duchêne, Fraenkel et al. regarding extensions of Wythoff game preserving its set of $\mathcal{P}$-positions. We also state a new conjecture about redundant moves in a variation of the game proposed by Fraenkel.

We assume that the reader has some knowledge about numeration systems and combinatorial game theory. See, for instance, [30] for a link between these two topics. About numeration systems and formal languages, we refer to [29].

2 Wythoff's game

This is the opportunity to recall some notions. For a survey on the many aspects of Wythoff's game, see [10]. Let us consider 2-player take-away games played on two heaps of tokens (it can be readily generalized to any finite number of heaps). A *position* is a pair of non-negative integers corresponding to the number of tokens on the two heaps. In *normal convention*, the first player unable to move loses the game (equivalently, the one taking the last token wins). On each turn, the current player has to remove at least one token (he/she may not pass). A *move* is a pair of non-negative integers corresponding to the number of tokens removed from the two heaps; a move (m_1, m_2) may be applied to a position (p_1, p_2) provided that $p_i \geq m_i$ for $i = 1, 2$. A game is then defined by its set of allowed moves.

Definition 2.1. A *game* is given by a function $G : \mathbb{N}^2 \to 2^{\mathbb{N}^2}$ that maps every position $p = (p_1, p_2)$ to the set of moves that can be chosen from p by the player. The set of *options* from (p_1, p_2) is $\{(p_1 - i, p_2 - j) \mid (i, j) \in G(p_1, p_2)\}$. A game is *impartial* if there is a unique function G for the two players (i.e., the set of moves available from any given position is the same for both players).

Wythoff's game [36], a modification of the game of Nim, is a classical 2-player take-away game described by the map

$$G_{\text{Wythoff}} : (p, q) \mapsto \{(i, 0) \mid i \in \llbracket 1, p\rrbracket\} \cup \{(0, j) \mid j \in \llbracket 1, q\rrbracket\} \cup \{(k, k) \mid k \in \llbracket 1, \min\{p, q\}\rrbracket\}.$$

With a game is associated a graph whose vertices are positions and edges correspond to moves. More precisely, there is an edge from position (p, q) to position $(p - i, q - j)$ if and only if the move (i, j) belongs to $G(p, q)$. If this graph is acyclic, then the game is said to be *acyclic*.

Definition 2.2. A position is a *$\mathcal{P}$-position* if there exists a strategy for the previous player (i.e., the player who will play on the next round) to win the game, whatever the move of the current player is. We let $\mathcal{P}(G)$, or simply $\mathcal{P}$, denote the set of $\mathcal{P}$-positions of the game G. Conversely, it is an *$\mathcal{N}$-position* if there exists a winning strategy for the first player (i.e., the one who is making the current move).

The characterization of the set of $\mathcal{P}$-positions of an impartial acyclic game is well-known. In other words, the set $\mathcal{P}(G)$ is a kernel of the game graph and acyclic graphs have a unique kernel [3].

Proposition 2.3 (Folklore). *The sets of $\mathcal{P}$- and $\mathcal{N}$-positions of an impartial acyclic game are uniquely determined by the following two properties:*

1. *Every move from a $\mathcal{P}$-position leads to an $\mathcal{N}$-position; equivalently there is no move between two $\mathcal{P}$-positions (stability property of $\mathcal{P}(G)$).*
2. *From every $\mathcal{N}$-position, there exists a move leading to a $\mathcal{P}$-position (absorbing property of $\mathcal{P}(G)$).*

Since we are providing some necessary background, we will encounter the concept of *Grundy function* g of an acyclic game-graph. It is recursively defined as follows.

Definition 2.4. For any position x, $g(x)$ is equal to $\text{MeX}\{g(y) \mid y \text{ is an option from } x\}$, where MeX denotes the *minimal excluded value*, i.e., the least non-negative integer not belonging to the set. In particular, $\text{MeX}\,\emptyset = 0$.

Since the game-graph is acyclic, sinks have Grundy value equal to zero (but other vertices may have a zero value). It is well-known that $\mathcal{P}$-positions of such a game are characterized by a zero Grundy value. See, for instance, [3, 30].

Fraenkel has obtained a characterization of the $\mathcal{P}$-positions of Wythoff's game using a non-standard numeration system [16]. Let $(F_i)_{i\geq 0} = 1, 2, 3, 5, 8, \ldots$ be the Fibonacci sequence whose first two terms are $1, 2$ and satisfying $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. As observed by Zeckendorf, any integer $n \geq 1$ can be decomposed as a sum of non-consecutive Fibonacci numbers [37]. Hence, there exist $\ell \geq 0$ and $d_\ell, \ldots, d_0 \in \{0, 1\}$ such that

$$n = \sum_{i=0}^{\ell} d_i\, F_i$$

with $d_\ell = 1$ and, for all $i \geq 1$, if $d_i = 1$ then $d_{i-1} = 0$. The word $d_\ell \cdots d_0$ over $\{0, 1\}$ has no factor 11 and is starting with 1. It is the *Fibonacci representation* of n and is denoted by $\text{rep}_F(n)$. We set $\text{rep}_F(0)$ to be the empty word ε. Dealing with decision procedures, this numeration system has

n	$\text{rep}_F(n)$	n	$\text{rep}_F(n)$
0	ε	8	10000
1	1	9	10001
2	10	10	10010
3	100	11	10100
4	101	12	10101
5	1000	13	100000
6	1001	14	100001
7	1010	15	100010

Table 1: The Fibonacci representation of the first few integers.

two important features: the set of Fibonacci representations is a regular language. The DFA on the left in Fig. 1 accepts the language $0^* \text{rep}_F(\mathbb{N})$ (it is common and generally useful to allow leading zeroes in front of a representation). Moreover addition is also recognizable by automaton. There exists a DFA accepting triplets $(u_\ell \cdots u_0, v_\ell \cdots v_0, w_\ell \cdots w_0)$ of words over $\{0, 1\}$ of the same length such that $\sum_{i=0}^{\ell} (u_i + v_i)\, F_i = \sum_{i=0}^{\ell} w_i\, F_i$. The Fibonacci system is an instance of an Ostrowski system associated with the golden ratio, as discussed in Section 3. Building a DFA for addition will also be studied in Section 4.1.

We will prove the next result in an automated way.

Theorem 2.5 (Fraenkel [16]). *A pair (a, b) of integers such that $a \leq b$ is a $\mathcal{P}$-position of Wythoff's game if and only if $\text{rep}_F(a)$ ends with an even number of zeroes and $\text{rep}_F(b)$ is a left-shift of $\text{rep}_F(a)$, i.e., $\text{rep}_F(b) = \text{rep}_F(a)0$.*

Taking care of the symmetry on the two heaps of tokens , we enter the following Walnut commands to build a binary predicate ppos for the set of $\mathcal{P}$-positions of Wythoff's game. We make sure to have a symmetric set of $\mathcal{P}$-positions.

```
reg end_even_zeros msd_fib "0*(00|0*1)*":
reg left_shift {0,1} {0,1} "([0,0]|([0,1][1,1]*[1,0]))*":
def ppos_asym "?msd_fib $end_even_zeros(a) & $left_shift(a,b)":
def ppos "?msd_fib $ppos_asym(a,b) | $ppos_asym(b,a)":
```

The resulting DFA is depicted in Fig. 2. An instruction such as `eval test "$ppos(3,5)":`

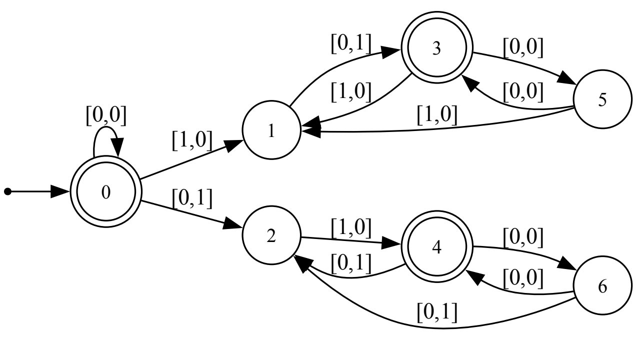

Figure 2: The DFA accepting $\mathcal{P}$-positions of Wythoff's game written in the Fibonacci numeration system.

evaluates to TRUE and `eval test "$ppos(1,3)":` evaluates to FALSE. See Table 2 for the first $\mathcal{P}$-positions.

We can provide an automated proof of Theorem 2.5 making use of Proposition 2.3.

Alternate proof of Theorem 2.5. The binary predicate ppos has to define a stable subset of $\mathbb{N}^2$; we have to check that there is no Wythoff's move between two $\mathcal{P}$-positions (p, q) and (r, s). From the form of the moves, it is only relevant to consider the case where $p \geq r$ and $q \geq s$. Either $(p, q) = (r, s)$ or we have two distinct positions. In the latter situation, we cannot have $p = r$ and $q > s$ because otherwise a move $(0, q - s)$ on a single heap between the two positions is available. Similarly, we cannot have $q = s$ and $p > r$. Finally, if $p > r$ and $q > s$, removing the same amount of tokens on both heap should not be available.

```
eval w_stable "?msd_fib Ap,q,r,s (($ppos(p,q) & $ppos(r,s) & p >= r & q >= s)
=> ((p=r & q=s) | (p>r & q>s & p+s!=q+r)) )":
```

Similarly, we have to check that the binary predicate ppos has to be absorbing. We have to check that for any $\mathcal{N}$-position (p, q) there is a Wythoff's move leading to some $\mathcal{P}$-position (x, y). Note that since $\mathcal{P}$ and $\mathcal{N}$ form a partition of $\mathbb{N}^2$, $(p, q) \neq (x, y)$.

```
eval w_absorbing "?msd_fib Ap,q (~$ppos(p,q) => Ex,y
( x<=p & y<=q & $ppos(x,y) & (p+y=q+x | p=x | q=y) )) ":
```

Intermediate steps require respectively a 142-state and 100-state automaton (after minimizations) and both expressions evaluate to TRUE within a few milliseconds. □

2.1 Extensions preserving $\mathcal{P}$-positions and a long-standing conjecture

In [11], the authors study the problem of adding moves to Wythoff's game while preserving its set of $\mathcal{P}$-positions. In fact, the addition of extra moves does not alter the absorbing property, but it could compromise stability. Let us define a move as *forbidden* whenever it would permit to play between two $\mathcal{P}$-positions. The set of forbidden moves is thus

$$\mathcal{F} = \{(a,b) \in \mathbb{N}^2 \mid \exists (p,q),(r,s) \in \mathcal{P} : (r-a, s-b) = (p,q)\}. \tag{1}$$

Adding to the rule-set of Wythoff's game moves in $\mathbb{N}^2 \setminus \mathcal{F}$ does not modify the set of $\mathcal{P}$-positions (because the stability property of $\mathcal{P}$ is preserved). Otherwise stated, with the extended rule-set game

$$G_{\text{Add}} : (p,q) \mapsto \left(\mathbb{N}^2 \setminus \mathcal{F}\right) \cap \{(x,y) \mid x \le p \wedge y \le q\}$$

we have $\mathcal{P}(G_{\text{Wythoff}}) = \mathcal{P}(G_{\text{Add}})$. One has to be really careful when using subtraction in `Walnut` commands. So the binary predicate corresponding to $\mathcal{F}$ is easily defined:

```
def wythoff_forbidden "?msd_fib Ep,q,r,s
 ($ppos(p,q) & $ppos(r,s) & p<=r & q<=s & p+a=r & q+b=s)":
```

With notation from [11], a pair (a,b) belongs to $\mathcal{F}$ if and only if $W_{a,b} = 1$. So the infinite matrix

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 1 & \\ 0 & 1 & 0 & 1 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 0 & \\ \vdots & & & & & \ddots \end{pmatrix}$$

is coding the moves that cannot be added to G_{Wythoff} without modifying its set of $\mathcal{P}$-positions. Otherwise stated, W is the characteristic bidimensional infinite word of the predicate $\mathcal{F}$. The corresponding automaton obtained with `Walnut` and accepting the Fibonacci representations of these pairs (a,b) is depicted in Fig. 3

We briefly outline a conjecture about the morphic structure of W stated in [11], which can now be proven. The authors introduced a bidimensional shape-symmetric morphism Ψ defined over a 26-letter alphabet. The image of each letter by Ψ is a block of size either 1×1, 1×2, 2×1 or 2×2. Roughly, the idea is that the morphism generates the blocks at the same rate in each direction in such a way that no overlap nor hole is created, see [26]. We have decided not to reproduce the entirety of the morphism here. The reader may have a look at the first few iterations of Ψ in Fig. 4. For instance, observe in $\Psi^2(a)$ that $\Psi(b)$ is a 2×1 block and symmetrically with respect to the main diagonal, $\Psi(c)$ is a 1×2 block. Conditions of this kind ensure that we get a well-defined growing block (here, we get squares with Fibonacci numbers as length of a side). For details, we refer the reader to [30]. To get a characteristic word, a coding ν that maps the 26 letters on $\{0,1\}$ is also provided. Now that all the concepts have been introduced, we are able to state the result.

Theorem 2.6. *With notation from [11], the morphism Ψ and the coding ν generate exactly the matrix W, i.e.,* $\lim_{n\to\infty} \nu(\Psi^n(a)) = W$.

The authors of [11] were not able to prove this result because they did not have a full syntactical characterization of the Fibonacci representations of the elements in $\mathcal{F}$. Thanks to `Walnut`, we have turned the conjecture into a statement. We may compare the DFA associated with Ψ conjectured in [11, Fig. 5] and the one from Fig. 3. States x and h in [11, Fig. 5] are merged to state 5 and similarly states f and y are merged to state 4.

Remark 2.7. Independently of the aforementioned conjecture, one can associate in a canonical way with the DFA in Fig. 3, a bidimensional morphism mapping states to blocks of states of size 1×1, 1×2, 2×1 or 2×2. Then, a result of Maes can be used to test if this morphism is shape-symmetric [26].

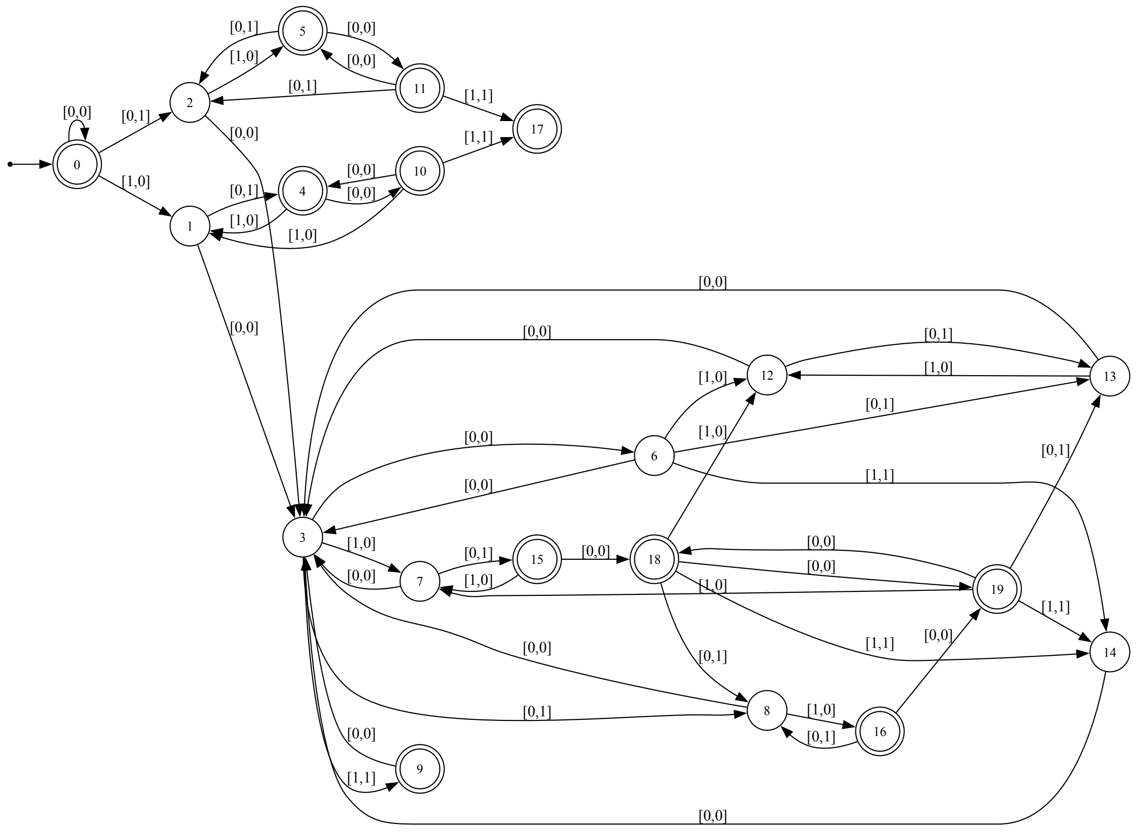

Figure 3: The DFA recognizing moves that cannot be adjoined without altering the set of $\mathcal{P}$-positions written in the Fibonacci numeration system.

$$\Psi(a) = \begin{array}{|cc|}\hline a & b \\ c & d \\ \hline\end{array}\,,\ \Psi^2(a) = \begin{array}{|cc|c|}\hline a & b & e \\ c & d & f \\ \hline e & h & i \\ \hline\end{array}\,,\ \Psi^3(a) = \begin{array}{|cc|c|cc|}\hline a & b & e & j & k \\ c & d & f & l & m \\ \hline e & h & i & g & b \\ \hline j & k & z & i & n \\ l & m & c & o & d \\ \hline\end{array}$$

Figure 4: First iterates of Ψ.

$$\Psi(a) = \begin{array}{|cc|}\hline 0 & 0 \\ 0 & 0 \\ \hline\end{array}\,,\ \Psi^2(a) = \begin{array}{|cc|c|}\hline 0 & 0 & 0 \\ 0 & 0 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array}\,,\ \Psi^3(a) = \begin{array}{|cc|c|cc|}\hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline\end{array}$$

Figure 5: Projection by ν of the first iterates of Ψ.

Let us make a side note. It is usual to let (A_n, B_n) denote the nth $\mathcal{P}$-position where $A_n < B_n$ and ordering the pairs by their first component. In [11], the following result is proved using a density argument and the algebraic characterization of (A_n, B_n) as $(\lfloor n\varphi \rfloor, \lfloor n\varphi^2 \rfloor)$ where φ is the Golden mean.

Proposition 2.8 (Prop. 12 in [11]). *We have*

$$\{(A_j - A_i, B_j - B_i) \mid j > i \geq 0\} = \{(A_n, B_n) \mid n > 0\} \cup \{(A_n + 1, B_n + 1) \mid n > 0\}.$$

Proof. This can be automated by the commands testing both inclusions

```
eval test_prop1 "?msd_fib Ap,q,r,s ( ($ppos(p,q) & $ppos(r,s) & p>q & r>s & p>r)
 => (Ex,y ($ppos(x,y) & ((p=r+x & q=s+y) | (p=r+x+1 & q=s+y+1))))) ":
eval test_prop2 "?msd_fib Ap,q (($ppos(p,q) & p!=0) => Ea,b,c,d
 ($ppos(a,b) & $ppos(c,d) & c>a & ((p+a=c & q+b=d)|(p+1+a=c & q+1+b=d))))":
```

which evaluates to TRUE and requires resp. a 351-state and a 256-state automaton. □

Remark 2.9. Consider the first few values of A_n and B_n given in Table 2. From [11, Prop. 9], we

n	0	1	2	3	4	5
A_n	0	1	3	4	6	8
B_n	0	2	5	7	10	13

Table 2: The first few $\mathcal{P}$-positions of Wythoff's game.

know that $A_n - 1 = \text{val}_F(\text{rep}_F(n-1)0)$, for all $n \geq 1$, i.e., looking at Fibonacci representations, $A_n - 1$ is the left-shift of $n - 1$. Similarly, $B_n - 2 = \text{val}_F(\text{rep}_F(A_n - 1)0)$, hence $B_n - 2$ can be obtained by shifting twice $n - 1$. We define two applications of the left-shift and thus two binary predicates

```
reg left_shift msd_fib msd_fib "([0,0]|([0,1][1,1]*[1,0]))*";
def left_shift2 "?msd_fib Eb $left_shift(a,b) & $left_shift(b,c)":
def an "?msd_fib $left_shift(n-1, a-1)| (a=0 & n=0)":
def bn "?msd_fib $left_shift2(n-1, b-2)| (b=0 & n=0)":
```

A crucial thing to remember when using Walnut is that the order of the arguments when a predicate is invoked is in alphabetical order of the free variables that appear in it when it is defined (see [33, p. 99]). Hence the predicate an (resp. bn) corresponds to the set of pairs (A_n, n) (resp. (B_n, n)). This means that the sequences $(A_n)_{n\geq 0}$ and $(B_n)_{n\geq 0}$ are Fibonacci-synchronized in the sense of Carpi and Maggi [7, 31]. Here is an alternative definition of the $\mathcal{P}$-positions of Wythoff's game.

```
def ppos_asym "?msd_fib En $an(a,n) & $bn(b,n)":
def ppos "?msd_fib $ppos_asym(a,b) | $ppos_asym(b,a)":
```

The following result was previously proved [11] using classical arguments.

Proposition 2.10 ([11, Cor. 1]). *A pair (i, j) of positive integers belongs to $\mathcal{F}$ if and only if one of the following three properties holds*

1. $(\text{rep}_F(i-1), \text{rep}_F(j-1)) = (u0, u01)$;
2. $(\text{rep}_F(i-2), \text{rep}_F(j-2)) = (u0, u01)$; *or*
3. $(\text{rep}_F(j - A_i - 2), \text{rep}_F(j - A_i - 2 + i)) = (u1, v0)$,

where u, v are valid F-representations in $\{0,1\}^*$.

Here is the corresponding Walnut proof.

Proof. Consider the following three commands.

```
def prop1 "?msd_fib Eu $left_shift(u,i-1) & $left_shift2(u,j-2)":
def prop2 "?msd_fib Eu $left_shift(u,i-2) & $left_shift2(u,j-3)":
def prop3 "?msd_fib Eu,v,a $an(a,i) & $left_shift2(u,j-a-3) & $left_shift(v,j-a-2+i)":
```

Note that $u1$ is a valid F-representation so the word u ends with a zero or is the empty word. This explains why we use `left_shift2` in the formula `prop3`. The F-representation of $j - A_i - 3$ has to end with two zeroes. We conclude the proof with the following two commands. The last one evaluates to TRUE.

```
def conjunction_prop "?msd_fib $prop1(i,j) | $prop1(j,i) |
   $prop2(i,j) | $prop2(j,i) | $prop3(i,j) | $prop3(j,i) | (i = 0 & j = 0)":

eval forbidden_characterization "?msd_fib Ai,j
 $conjunction_prop(i,j) <=> $wythoff_forbidden(i,j)":
```

The last command evaluating to TRUE completes the proof. □

2.2 Redundant moves

Definition 2.11. A move is *redundant* if the set of $\mathcal{P}$-positions is unchanged when the move is removed from the rule-set.

In [11], it is shown that Wythoff's game has no redundant moves. As observed in [22] a move $m = (m_1, m_2)$ is not redundant if there exists a $\mathcal{N}$-position (p, q) such that m is the unique winning move from (p, q) to some $\mathcal{P}$-position. We can again determine these moves using the following three Walnut commands. They express that each of the three types of moves of Wythoff's game $(i, 0)$, $(0, i)$ and (i, i) respectively are not redundant.

```
eval non_redundant1 "?msd_fib Ai (i>0 => (Ep,q (~$ppos(p,q) & $ppos(p-i,q) &
(Aj ((j<=p & j!=i) => ~$ppos(p-j,q)) ) & (Aj (j<=q => ~$ppos(p,q-j)) ) &
(Aj ((j<=p & j<=q) => ~$ppos(p-j,q-j))))))":

eval non_redundant2 "?msd_fib Ai (i>0 => (Ep,q (~$ppos(p,q) & $ppos(p,q-i) &
(Aj ((j<=q & j!=i) => ~$ppos(p,q-j)) ) & (Aj (j<=p => ~$ppos(p-j,q)) ) &
(Aj ((j<=p & j<=q) => ~$ppos(p-j,q-j))))))":

eval non_redundant3 "?msd_fib Ai (i>0 => (Ep,q ( ~$ppos(p,q) & $ppos(p-i,q-i) &
(Aj (j<=q => ~$ppos(p,q-j))) & (Aj (j<=p => ~$ppos(p-j,q))) &
(Aj ((j<=p & j<=q & j!=i) => ~$ppos(p-j,q-j))))))":
```

For each of them, the formulas evaluates to TRUE. The first two formulas are symmetric and require at most 136 states at some intermediate computation. The last one requires up to 223 states.

2.3 $\mathcal{R}$-Wythoff

In [22] the following restriction of Wythoff's game has been considered by Nhan Bao Ho. Each move is either to remove a positive number of tokens from the larger heap (or any heap if the two heaps are the same size) or to remove the same number of tokens from both heaps. Note that if the sizes of the two heaps are not equal, then removing tokens from the smaller heap is not allowed:

$$G_{\text{R-Wythoff}} : (a, b) \mapsto \begin{cases} \{(i, 0) \mid i \in [\![1, a]\!]\} \cup \{(k, k) \mid k \in [\![1, b]\!]\}, & \text{if } a > b; \\ \{(0, j) \mid i \in [\![1, b]\!]\} \cup \{(k, k) \mid k \in [\![1, a]\!]\}, & \text{if } b > a; \\ \{(0, i), (i, 0), (i, i) \mid i \in [\![1, a]\!]\} & \text{if } a = b. \end{cases}$$

This is a *restriction* because $G_{\text{Wythoff}}(a, b) \supset G_{\text{R-Wythoff}}(a, b)$ for all (a, b). Note that it is a *variant* rule-set because the moves depend on the position. Recall that a game is *invariant* if the set of options is the same for all positions (provided enough tokens are available). In this section we show that many of the results about $G_{\text{R-Wythoff}}$ can be automatically proved using `Walnut` and thus avoiding a somewhat tedious case analysis. This is because the set of remaining moves can still be defined in first-order logic.

We define the options (c, d) that are accessible from a given position (a, b) as follows. Formally, we define the set of 4-tuples (a, b, c, d) such that there is an allowed move from (a, b) to (c, d).

```
def options_rwythoff "?msd_fib
  (b<a => Ex (x>0 & ((a=c+x & b=d+x) | (a=c+x & b=d)))
& (a<b => Ex (x>0 & ((a=c+x & b=d+x) | (a=c & b=d+x))))
& (a=b => Ex (x>0 & ((a=c+x & b=d+x) | (a=c& b=d+x) | (a=c+x & b=d))))":
```

Theorem 2.12 ([22, Thm. 2.2])**.** *The $\mathcal{R}$-Wythoff game preserves the $\mathcal{P}$-positions of Wythoff's game, i.e., the $\mathcal{P}$-positions of $\mathcal{R}$-Wythoff are identical to those of Wythoff's game.*

Proof. Recall that we have built a binary predicate `ppos` for the set of $\mathcal{P}$-positions of Wythoff's game. Thanks to Proposition 2.3 it is therefore enough to evaluate the following two commands

```
eval rwythoff_stable "?msd_fib Ap,q,r,s ( ($ppos(p,q) & $ppos(r,s) & p>=r & q>=s)
=> ((p=r & q=s) | ~$options_rwythoff(p,q,r,s)) )":
```

and

```
eval rwythoff_absorbing "?msd_fib Ap,q (~$ppos(p,q)
=> Ex,y ( x<=p & y<=q & $ppos(x,y) & $options_rwythoff(p,q,x,y) )) ":
```

They both return `TRUE`. This shows that the set of $\mathcal{P}$-positions of Wythoff's game is stable and absorbing for the rule-set of $\mathcal{R}$-Wythoff. □

Theorem 2.13 ([22, Thm. 2.3])**.** *There is no restriction of $\mathcal{R}$-Wythoff preserving its $\mathcal{P}$-positions.*

Proof. We use the same strategy as in Section 2.2. The three commands

```
eval rwythoff_non_redundant1 "?msd_fib Ai i > 0 =>
 (Ep,q $options_rwythoff(p,q,p-i,q) & ~$ppos(p,q) & $ppos(p-i,q) &
 (Ar,s (r <= p & s <= q & $options_rwythoff(p,q,r,s) & $ppos(r, s))
=> (r+i=p & s=q)))":
```

and

```
eval rwythoff_non_redundant2 "?msd_fib Ai i > 0 =>
 (Ep,q $options_rwythoff(p,q,p,q-i) & ~$ppos(p,q) & $ppos(p,q-i) &
 (Ar,s (r <= p & s <= q & $options_rwythoff(p,q,r,s) & $ppos(r, s))
=> (r=p & s+i=q)))":
```

and

```
eval rwythoff_non_redundant3 "?msd_fib Ai i > 0 =>
 (Ep,q $options_rwythoff(p,q,p-i,q-i) & ~$ppos(p,q) & $ppos(p-i,q-i) &
 (Ar,s (r <= p & s <= q & $options_rwythoff(p,q,r,s) & $ppos(r, s))
=> (r+i=p & s+i=q)))":
```

evaluate to `TRUE`. □

We now turn to the characterization of positions with Grundy value equal to 1.

Lemma 2.14. *Let* $G = (V, E)$ *be a directed acyclic graph with Grundy function* g*. Let* G' *be the subgraph of* G *induced by the set of vertices* $V \setminus \{v \in V \mid g(v) = 0\}$ *and* h *be the Grundy function of* G'*. For all vertices* v *of* G'*, we have* $g(v) = h(v) + 1$*. In particular, the set of vertices* v *of* G *such that* $g(v) = 1$ *is the kernel of* G'*.*

Proof. Let B_0 be the kernel of G, i.e., $B_0 = \{v \in V \mid g(v) = 0\}$. Let E' be the set of edges of G'. Let $v \in V \setminus B_0$. By definition of h, we have $h(v) = \text{MeX}\{h(y) \mid (v, y) \in E'\}$. Hence

$$\begin{aligned} h(v) + 1 &= \text{MeX}\left(\{0\} \cup \{h(y) + 1 \mid (v, y) \in E'\}\right) \\ &= \text{MeX}\left(\{g(y) \mid y \in B_0 \text{ and } (v, y) \in E\} \cup \{h(y) + 1 \mid (v, y) \in E'\}\right). \end{aligned}$$

Since B_0 is absorbing on G, this ensures that there exists some $y \in B_0$ such that $(v, y) \in E$. Now define a function $\alpha : V \to \mathbb{N}$ by $\alpha(v) = 0$ if $v \in B_0$ and $\alpha(v) = h(v) + 1$ otherwise. Hence, for all $v \in V \setminus B_0$, $\alpha(v) = \text{MeX}\{\alpha(y) \mid (v, y) \in E\}$. Otherwise stated, α is the Grundy function of G. □

Theorem 2.15 ([22, Thm. 2.4])**.** *In* $\mathcal{R}$*-Wythoff, the position* (a, b) *with* $a \le b$ *has Grundy value* 1 *if and only if* (a, b) *is an element of the set* $C = \{(2, 2), (4, 6), (\lfloor n\varphi \rfloor - 1, \lfloor n\varphi \rfloor + n - 1) \mid n \ge 1, n \ne 2\}$*.*

Proof. We make use of Remark 2.9 where are defined the binary predicates an and bn. Note that $(\lfloor n\varphi \rfloor - 1, \lfloor n\varphi \rfloor + n - 1) = (\lfloor n\varphi \rfloor - 1, \lfloor n\varphi^2 \rfloor - 1) = (A_n - 1, B_n - 1)$ and $a = A_n - 1$ if and only if `an(a+1,n)` holds. So we may define a predicate for the set C given in the statement:

```
def rwythoff_setC_asym "?msd_fib
 (a=2 & b=2) | (a=4 & b=6) | (En n!=2 & n>=1 & $an(a+1,n) & $bn(b+1,n))":
def rwythoff_setC
 "?msd_fib $rwythoff_setC_asym(a,b) | $rwythoff_setC_asym(b,a)":
```

It remains to check that this candidate C is indeed the right one. We make use of Lemma 2.14. The set B_1 of positions with Grundy value 1 is characterized by the following three properties and we check that there are satisfied by C

- $B_1 \cap \mathcal{P} = \emptyset$,

  ```
  eval rw_valid_grundy1a "?msd_fib Ap,q $rwythoff_setC(p,q)
       => (~$ppos(p,q))":
  ```

- there is no move between any two positions in B_1,

  ```
  eval rw_valid_grundy1b "?msd_fib Ap,q $rwythoff_setC(p,q)
       => (Ar,s $rwythoff_setC(r,s) => ~$options_rwythoff(p,q,r,s))":
  ```

- every position not in $B_1 \cup \mathcal{P}$ has an option in B_1.

  ```
  eval rw_valid_grundy1c "?msd_fib (Ar,s (~$rwythoff_setC(r,s) & ~$ppos(r,s))
       => Ep,q $rwythoff_setC(p,q) & $options_rwythoff(r,s,p,q))":
  ```

These three expressions again evaluate to `TRUE`. This proves that $C = B_1$. □

Remark 2.16. Note that in the above statement and its proof, we are lucky to have a candidate for the set of positions with Grundy value 1 and moreover, this set has a regular representation (in the Fibonacci system). Having this candidate, we can therefore make use of `Walnut`. So a similar approach may be used each time these two properties are satisfied.

2.4 $\mathcal{E}$-Wythoff

Nhan Bao Ho also introduces an extension of Wythoff's game obtained by adjoining a move removing k tokens from the smaller heap (or any heap if the two heaps have the same size) and ℓ tokens from the other heap where $\ell < k$. Options (c, d) available from (a, b) in this variation are coded by the following 4-ary predicate.

```
def options_ewythoff "?msd_fib
 (a=c & d<b) | (b=d & c<a) | (a+d = b+c & c<a & d<b)
 | (b<=a & c<a & d<b & b+c>a+d) | (a<=b & c<a & d<b & a+d>b+c)":
```

We may thus duplicate the proof of Theorem 2.12 and get the following result.

Theorem 2.17 ([22, Thm. 3.1]). *The $\mathcal{E}$-Wythoff game preserves the $\mathcal{P}$-positions of Wythoff's game, i.e., the $\mathcal{P}$-positions of $\mathcal{E}$-Wythoff are identical to those of Wythoff's game.*

Theorem 2.18 ([22, Thm. 3.2]). *In $\mathcal{E}$-Wythoff, the position (a, b) with $a \leq b$ has Grundy value 1 if and only if (a, b) is of the form $(\lfloor n\varphi \rfloor - 1, \lfloor n\varphi \rfloor + n - 1)$ for some $n \geq 1$.*

We can duplicate the proof of Theorem 2.15, we simply have to consider the following definition.

```
def ewythoff_grundy1_asym "?msd_fib En n >= 1 & $an(a+1,n) & $bn(b+1,n)":
```

3 Fraenkel's variation and Ostrowski systems

Fraenkel proposed a variation of Wythoff's rule: Instead of taking the same number of tokens on both heaps, one may remove $k > 0$ tokens from one heap and $\ell > 0$ from the other one, provided that $|k - \ell| < m$ [16]. For $m = 1$, we get back to the classical Wythoff's game. Let us recall the notion of Ostrowski systems [1, 4].

For an irrational number $0 < \alpha < 1$ with continued fraction expansion $[d_0; d_1, d_2, \ldots]$ (where $d_0 = 0$), we say that $p_i/q_i = [d_0; d_1, d_2, \ldots, d_i]$ is a convergent of α where the p_i and q_i satisfy the following relations:

$$p_{-2} = 0, \quad p_{-1} = 1, \quad p_i = d_i p_{i-1} + p_{i-2}, \tag{2}$$
$$q_{-2} = 1, \quad q_{-1} = 0, \quad q_i = d_i q_{i-1} + q_{i-2}. \tag{3}$$

Let $(q_i)_{i \geq 0}$ be the sequence denoting the denominator of the convergents of α. We have $q_0 = 1$ and $q_1 = d_1$. Then every non-negative integer n can be uniquely represented as

$$n = \sum_{i=0}^{\ell} a_i q_i,$$

where the a_i are integers satisfying the following three conditions:

- $0 \leq a_0 < d_1$
- $0 \leq a_i \leq d_{i+1}$, for $i \geq 1$; and
- for all $i \geq 1$,
$$\text{if } a_i = d_{i+1} \text{ then } a_{i-1} = 0. \tag{4}$$

Thus any integer $n \in \mathbb{N}$ may be represented uniquely in MSD-first notation as a word over the alphabet $\mathbb{N}$; we let $\text{rep}_\alpha(n) := a_\ell \cdots a_0$, where the a_i satisfy the above conditions. Note that whenever the coefficients d_i in the continued fraction expansion are bounded, say by M, then the representations of integers are over the finite alphabet $\{0, \ldots, M\}$.

Note that if $d_1 = 1$, then a_0 is equal to zero, so n can be represented as

$$n = \sum_{i=1}^{\ell} a_i q_i \tag{5}$$

with the same conditions as above. Similar developments can be considered with the numerator of the convergents. We may use the terminology of q-*system* or p-*system*.

Remark 3.1. About `Walnut` [2], the Ostrowski code that implements a q-system is assuming that a number between 0 and 1 is used. There is an implicit 0 at the beginning of the continued fraction that is ignored. In our game related problem, we deal with a real number larger than 1 but this is not an issue.

Let α be an irrational number such that $1 < \alpha < 2$ with continued fraction expansion $[d_0; d_1, d_2, \ldots]$ (where $d_0 = 1$). In that case, $1/\alpha < 1$ has the shifted expansion $[0; d_0, d_1, d_2, \ldots]$. Let $p_i^{(\alpha)}$ (resp. $q_i^{(1/\alpha)}$) be the numerator (resp. denominator) of the ith convergent of α (resp. $1/\alpha$). It is easy to check that $p_i^{(\alpha)} = q_{i+1}^{(1/\alpha)}$ for all $i \geq -2$. Since $d_0 = 1$, (5) can be written

$$n = \sum_{i=1}^{\ell} a_i\, q_i^{(1/\alpha)} = \sum_{i=1}^{\ell} a_i\, p_{i-1}^{(\alpha)}.$$

This means that, for $\alpha \in (1, 2)$, if we want to deal with the p-system associated with α when using `Walnut`, the q-system associated with $1/\alpha$ is used internally.

In what follows, we consider the case of Fraenkel's game with $m = 2$ but the discussion can be made for any natural number $m \geq 1$.

To get a characterization of the $\mathcal{P}$-positions of this game, we need a particular Ostrowski system based on the quadratic irrational $\frac{2-m+\sqrt{4+m^2}}{2}$, which has the continued fraction expansion $[1, \overline{m}] = [1; m, m, m, \ldots]$. So consider the Ostrowski numeration system based on $\alpha = [1, \overline{2}] = \sqrt{2}$ and the set of moves (we take extra care to deal with non-negative results). These are defined as follows in `Walnut`:

```
ost ost2 [1] [2]:
def ost2_move "?msd_ost2 (a+b>0) & (a=0 | b=0 | (a>=b & a<b+2) | (a<b & b<a+2))";
```

In view of Remark 3.1, let $(p_n)_{n\geq 0}$ be the sequence of numerators of the convergents of the continued fraction α. This sequence starts with $p_0 = 1$, $p_1 = m + 1$ and $p_{n+2} = m\, p_{n+1} + p_n$ for all $n \geq 0$. In particular, for $m = 2$, the first few terms are $1, 3, 7, 17, 41, 99, 239, 577, 1393, 3363, \ldots$. The valid expansions in this p-system are words over $\{0, 1, 2\}$ with the only restriction that a 2 can only be followed by a 0. See the central DFA in Fig. 1.

Fraenkel obtains a characterization of the $\mathcal{P}$-positions (A_n, B_n) similar to the one of Theorem 2.5 except that they are represented in the corresponding Ostrowski system: the p-representation of A_n ends with an even number of zeroes, the p-representation of B_n is a left-shift of the one of A_n [16, §5]. We can therefore use the following commands

```
reg end_even_zeros msd_ost2 "0*(00|0*[1-2])*";
def ost2_ppos_asym "?msd_ost2 $end_even_zeros(a) & $ternary_shift(a,b)":
def ost2_ppos "?msd_ost2 $ost2_ppos_asym(a,b) | $ost2_ppos_asym(b,a)":
```

where we have encoded[2] a DFA for a left-shift over a 3-letter alphabet as seen in Fig. 6.

To get a proof of Fraenkel's characterization (for a fixed value of the parameter m), we proceed as in the previous situations. The following two commands evaluate to `TRUE`.

```
eval ost2_stable "?msd_ost2 Ap,q,r,s ($ost2_ppos(p,q) & $ost2_ppos(r,s)
                                     & r<=p & s<=q) => ~$ost2_move(p-r,q-s)":
eval ost2_absorbing "?msd_ost2 Ap,q (~$ost2_ppos(p,q) =>
              (Ex,y (x<=p & y<=q & $ost2_ppos(p-x,q-y) & $ost2_move(x,y))))":
```

[2]and stored in the `Automata Library`

```
{0,1,2} {0,1,2}

0 1
0 0 -> 0
0 1 -> 1
0 2 -> 2

1 0
1 0 -> 0
1 1 -> 1
1 2 -> 2

2 0
2 0 -> 0
2 1 -> 1
2 2 -> 2
```

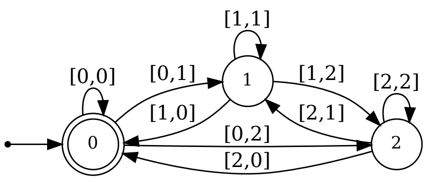

Figure 6: The left-shift DFA over $\{0, 1, 2\}$

3.1 Detecting redundant moves

We may also look for redundant moves (this question was not treated in [16]). The situation here proves to be more interesting than in the previous sections where there were no redundant moves at all for the considered games of Wythoff and $\mathcal{R}$-Wythoff. We will show that there are several redundant moves. However, it is not straightforward to remove them simultaneously while maintaining a game (with a restricted set of rules) that has the same set of $\mathcal{P}$-positions. First proceed as in Section 2.2. We remark that this is the first time in this paper that `Walnut` requires considerable resources but that are still manageable on a standard laptop; constructing the automaton for the predicate `ost2_non_redundant` below requires approximately one hundred seconds and some intermediate constructions with 2522 states requiring up to 7GB of RAM.

```
def ost2_non_redundant "?msd_ost2 $ost2_move(a,b)
  & Ep,q (~$ost2_ppos(p,q) & $ost2_ppos(p-a,q-b)
  & (Ac,d((a!=c|b!=d) & $ost2_move(c,d) & c<=p & d<=q)=> ~$ost2_ppos(p-c,q-d)))":
def ost2_redundancies "?msd_ost2 $ost2_move(a,b) & ~$ost2_non_redundant(a,b)":
```

So having a move (a, b) satisfying the predicate `ost2_redundancies` means that whenever a player may choose the move (a, b) in a winning strategy, then an alternative (depending on the actual $\mathcal{N}$-position) always exists.

The DFA in Fig. 7 accepts the redundant moves expressed in the convenient $\sqrt{2}$-Ostrowski system. From the definition of the game, it is clear that the set of redundant moves is symmetric. So to ease the presentation, we only represent moves (a, b) satisfying $a \leq b$.

The analysis of the automaton in Fig. 7 reveals three states $0, 2, 4$ corresponding to valid representations in the numeration system under consideration. A move (a, b) with $a \leq b$ is redundant if and only if the pair $(\mathrm{rep}_{\sqrt{2}}(a), \mathrm{rep}_{\sqrt{2}}(b))$ is of the form $(u0(20)^n, u1(00)^n)$, $(u0(20)^n2, u1(00)^n0)$, $(u1(20)^n, u2(00)^n)$ or $(u1(20)^n2, u2(00)^n0)$ for convenient prefix u. The extra states $1, 3, 5$ handle the small values for which $(0, 1)$, $(1, 2)$ and $(2, 3)$ are not redundant. If $\mathrm{rep}_{\sqrt{2}}(a)$ and $\mathrm{rep}_{\sqrt{2}}(b)$ have the same length, then state 2, 3 or 4 is reached. Otherwise, state 1 is reached first. It is also interesting to observe how the carry propagation operates in this system. Indeed, adding one to a number represented by $u0(20)^n$ (resp. $v1(20)^n$) gives the representation $u1(00)^n$ (resp. $v2(00)^n$) assuming that u does not end with 2. There is no accepting path of the form (w, w) thus all moves (n, n) are non-redundant. The following command evaluates to `TRUE`:

```
eval ost2_characterize "?msd_ost2 Aa,b (a<=b) =>
     (($ost2_redundancies(a,b) |(a=0&b=1)|(a=1&b=2)|(a=2&b=3)) <=> b=a+1)":
```

Consequently, we have obtained the following result.

Proposition 3.2. *The variation of Wythoff's game where one is allowed to remove $k > 0$ tokens from one heap and $\ell > 0$ from the other one, provided that $|k - \ell| < 2$, has infinitely many redundant moves*

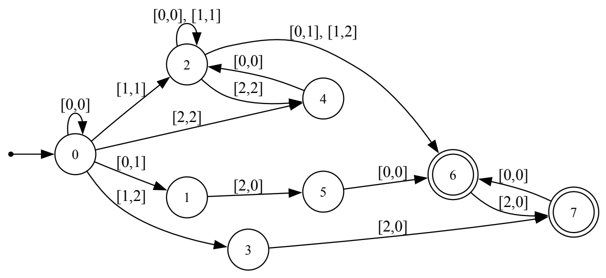

Figure 7: The DFA recognizing the redundant moves (a, b) with $a \leq b$ ($m = 2$).

whose representations in the corresponding p*-system are given in Fig. 7. Precisely, the redundant moves are exactly the pairs* $(n, n+1)$ *and* $(n+1, n)$ *for all* $n \geq 3$.

For any value of the parameter m, moves of the form $(0, n)$ and $(n, 0)$ are non-redundant for all $n > 0$. Indeed, from the $\mathcal{N}$-position $(0, n)$, there is a single move leading to $(0, 0)$ and consisting in the removal of all tokens. In what follows, we therefore do not consider these moves.

Remark 3.3. We characterized in the same way the set of redundant moves for $m = 3$ and $m = 4$. For the corresponding commands, we needed a laptop with more resources. For $m = 3$, computing the automaton `ost3_non_redundant` took around 150 seconds and required approximately 27GB of RAM. For $m = 4$, we needed already approximately 45GB of RAM and took 21 minutes. Since the following checks evaluate to `TRUE`, we get a result similar to Proposition 3.2: for $m = 3$, the redundant moves are $(n, n+i)$, $(n+i, n)$ for $i = 1, 2$ and $n \geq 5-i$. For $m = 4$, the redundant moves are $(n, n+i)$, $(n+i, n)$ for $i = 1, 2, 3$ and $n \geq 6-i$.

```
eval test1 "?msd_ost3 An (n>=4)<=>$ost3_redundancies(n,n+1)":
eval test2 "?msd_ost3 An (n>=3)<=>$ost3_redundancies(n,n+2)":
eval test3 "?msd_ost3 An ~$ost3_redundancies(n,n)":
```

```
eval test1 "?msd_ost4 An (n>=5)<=>$ost4_redundancies(n,n+1)":
eval test2 "?msd_ost4 An (n>=4)<=>$ost4_redundancies(n,n+2)":
eval test3 "?msd_ost4 An (n>=3)<=>$ost4_redundancies(n,n+3)":
eval test4 "?msd_ost4 An ~$ost4_redundancies(n,n)":
```

Even though the case $m = 4$ requires a machine with more resources than an average standard computer (at the time of writing), this tends to show that `Walnut` can be used in this context to formulate new conjectures (for an arbitrary parameter m), which we do now:

Conjecture 3.4. *Let* $m \geq 2$*. The set of redundant moves of the variation of Wythoff's game where one is allowed to remove* $k > 0$ *and* $\ell > 0$ *provided that* $|k - \ell| < m$ *is*

$$\bigcup_{1 \leq i < m} \{(n, n+i), (n+i, n) \mid n \geq m - i + 2\}.$$

The conjecture holds for $m = 2$, 3, and 4 by the previous remark. We were not able to get the automaton for redundant moves in case of $m = 5$ even with a laptop with 64GB of RAM. We explain below that an alternative exists to show that the conjecture holds, for instance, for $m = 5$.

Remark 3.5. Assuming access to a computer with only limited memory, it is still possible to evaluate some simpler commands (in which we have removed a quantifier) and thus test particular values. Such a procedure allowed us to test the above conjecture some more. We let the reader define the Ostrowski system `ost5`, a shift automaton over the alphabet $\{0, \ldots, 5\}$ and the commands `ost5_move` and `ost5_ppos` accordingly. For some fixed values x and y, we may run the following command:

```
eval test_conj "?msd_ost5 Ap,q ((~$ost5_ppos(p,q) & $ost5_ppos(p-x,q-y))
  => Ec,d $ost5_move(c,d) & $ost5_ppos(p-c,q-d) & (c!=x|d!=y))":
```

Such a command evaluates to TRUE for $(x, y) \in \{(3, 7), (4, 7), (5, 7), (6, 7)\}$ showing that these moves are redundant.

As a final comment about the conjecture, even though we do not have access to an automaton for `ost5_redundancies`, we were able to run four commands like the following ones on a standard laptop with 16GB of RAM.

```
eval test_conj1 "?msd_ost5 An (n>=6 => (Ap,q (p>=n & q>n &
   ~$ost5_ppos(p,q) & $ost5_ppos(p-n,q-n-1))
=> Ec,d ((c!=n|d!=n+1) & p>=c & q>=d & $ost5_move(c,d)& $ost5_ppos(p-c,q-d))))":
...
eval test_conj4 "?msd_ost5 An (n>=3 => (Ap,q (p>=n & q>n+3 &
   ~$ost5_ppos(p,q) & $ost5_ppos(p-n,q-n-4))
=> Ec,d ((c!=n|d!=n+4) & p>=c & q>=d & $ost5_move(c,d)& $ost5_ppos(p-c,q-d))))":
```

As Walnut returns `True` for each of the queries, we have shown that moves of the form $(n, n + i)$ are redundant for $1 \leq i < 5$ and $n \geq 7 - i$. Moreover, the next command, returning `True`, shows that moves of the form (n, n) are non-redundant.

```
eval test_conj5 "?msd_ost5 An (n>0 => Ep,q (p>=n & q>=n &
  ~$ost5_ppos(p,q) & $ost5_ppos(p-n,q-n)
& ~(Ec,d ((c!=n|d!=n) & p>=c & q>=d & $ost5_move(c,d)& $ost5_ppos(p-c,q-d)))))":
```

From a practical perspective, the last part of this formula is logically equivalent to

```
Ac,d ((c!=n|d!=n) & p>=c & q>=d & $ost5_move(c,d))=> ~$ost5_ppos(p-c,q-d)
```

but the way the formulas are handled by `Walnut` makes it harder to evaluate. Much more memory is required. To conclude with the proof of the conjecture for $m = 5$, it remains to check only that the short "non-diagonal" moves $(i, i + j)$ are non-redundant with $1 \leq j \leq 4$ and $i + j \leq 6$. A formula is easy to produce for each candidate move and their validity can be verified with `Walnut` individually. There is a way to do these individual checks with the `load` command provided by `Walnut`: each of the queries can be written in a text file (in our case, in `walnut_games_commands.txt` on separate lines and stored in the `Command Files` subfolder. The command `load walnut_games_commands.txt;` will then prompt `Walnut` to evaluate each command in the text file; in this case returning `True` for each of the queries. Thus we have shown that Conjecture 3.4 holds for $m = 5$ as well.

Remark 3.6. In [12], the authors consider games whose associated numeration system is based on an irrational number of the form $\alpha_k = [1; \overline{1, k}]$ where $k \geq 2$ is an integer parameter. The nth $\mathcal{P}$-position is given by $(\lfloor n\alpha_k \rfloor, \lfloor n\beta_k \rfloor)$ where $1/\alpha_k + 1/\beta_k = 1$. In the p-system associated with α_k, one may conjecture that A_n (resp. B_n) is represented in this system by the nth valid representation ending with an even (resp. odd) number of zeroes. The problem is that the p-representation of B_n is no longer the shift of the representation of A_n. There is no clear syntactical correspondence. To be able to use `Walnut`, the sequences (A_n) and (B_n) should be synchronized, i.e., there exist an automaton accepting pairs (n, A_n) (resp. (n, B_n)) written in the Ostrowski system. If that was the case, one could apply again a similar strategy as discussed in Remark 2.9.

3.2 Getting rid of redundant moves

The question now arises as to whether these redundant moves can be removed all at once without altering the set of $\mathcal{P}$-positions. As an example, for $m = 2$, assume that we remove the redundant moves $(4, 5)$ and $(16, 17)$ represented respectively by $(11, 12)$ and $(0202, 1000)$. By symmetry, let us also remove the moves $(5, 4)$ and $(17, 16)$. Then the set of $\mathcal{P}$-positions does not change because the following evaluates to `TRUE`.

```
def ost2_new2set "?msd_ost2 $ost2_move(a,b) & (a!=4|b!=5) & (a!=16|b!=17)
                                             & (a!=5|b!=4) & (a!=16|b!=17)":
eval ost2_new2set_absorbing "?msd_ost2 Ap,q (~$ost2_ppos(p,q) =>
        (Ex,y (x<=p & y<=q & $ost2_ppos(p-x,q-y) & $ost2_new2set(x,y))))":
```

However we could possibly encounter a situation where (p, q) is a $\mathcal{N}$-position and moves (a, b) such that $(p - a, q - b)$ is a $\mathcal{P}$-position all satisfy the predicate `~$ost2_non_redundant(a,b)`. Removing all these moves will therefore alter the absorbing property. Note that if such a position (p, q) exists, we cannot have a single winning move (a, b) because otherwise it would satisfy `$ost2_non_redundant(a,b)`. For $m = 2$, we find two redundant moves that may not be simultaneously removed without altering the set of $\mathcal{P}$-positions. The following command finds small $\mathcal{N}$-positions (p, q) whose winning moves are all redundant. It produces a DFA with 22 states.

```
def ost2_try_cannot_remove_two "?msd_ost2 (~$ost2_ppos(p,q) & p<20 & q<20
 & (Aa,b (a<=p & b<=q & $ost2_move(a,b) & $ost2_ppos(p-a,q-b))
                   => ~$ost2_non_redundant(a,b)))":
```

For instance, it recognizes the position $(11, 12)$. The following command shows that the only available moves are $(11, 12)$ and $(10, 9)$ represented respectively by $(111, 112)$ and $(110, 102)$ and thus accepted by the DFA in Fig. 7.

```
def ost2_which_moves "?msd_ost2 i<=11 & j<=12 & $ost2_ppos(11-i,12-j)
                                                & $ost2_move(i,j)":
```

Let us proceed as before. Remove these redundant moves $(11, 12)$ and $(10, 9)$. Now, the final commands evaluates to `FALSE` as expected.

```
def ost2_new2set "?msd_ost2 $ost2_move(a,b) & (a!=11|b!=12) & (a!=10|b!=9)";
eval ost2_new2set_absorbing "?msd_ost2 Ap,q (~$ost2_ppos(p,q) =>
        (Ex,y (x<=p & y<=q & $ost2_ppos(p-x,q-y) & $ost2_new2set(x,y))))":
```

There are indeed infinitely many pairs of redundant moves (a, b) and (c, d) such that suppressing these two moves alter the set of $\mathcal{P}$-positions. This can be checked by inspecting the automaton with 83 states resulting of the following command (needing 35GB of RAM) which contains infinitely many accepting paths (with different values).

```
eval ost2_test "?msd_ost2 ($ost2_redundancies(a,b) & $ost2_redundancies(c,d)
& (a!=c|b!=d)) & Ep,q (~$ost2_ppos(p,q) => (Ex,y (x<=p & y<=q & $ost2_move(x,y)
& $ost2_ppos(p-x,q-y))=> ((x=a&y=b)|(x=c&y=d))))":
```

4 Beyond Ostrowski systems

In [17] the game discussed in the previous section has been generalized. As for the game of Nim, one can remove a positive number of tokens from one of the two heaps. Or, one can remove k tokens from one heap and ℓ from the other one, provided that $0 < k \leq \ell < sk + m$ where s, m are two positive integer parameters. For $s = 1$, this is the game with parameter m discussed in the previous section and for $s = m = 1$, this is Wythoff's game.

As we've already seen, considering a suitable numeration system allows one to get a syntactical characterization of the $\mathcal{P}$-positions of the game and Fraenkel's theorem for Wythoff's game can be generalized one step further. Here, the associated numeration system is based on the linear recurrence sequence $(U_n)_{n\geq 0}$ with characteristic polynomial

$$X^2 - (s + m - 1)X - s \tag{6}$$

and initial conditions $U_0 = 1$ and $U_1 = s + m$. This polynomial is the minimal polynomial of a Pisot number β: for $s, m \geq 1$, the root β is larger than 1 and the other one has a modulus less than 1. Note that the numeration system with parameters $s = 2$ and $m = 1$ was recently discussed in [35] and also in [14] in relation with the morphism $a \mapsto aab$ and $b \mapsto aa$.

Lemma 4.1. *Let* $s, m \geq 1$ *and* $(U_i)_{i\geq 0}$ *be the sequence defined by* $U_0 = 1$, $U_1 = s+m$ *and, for all* $n \geq 2$, $U_n = (s+m-1)U_{n-1} + sU_n$. *Any integer* $n > 0$ *has a unique representation* $d_\ell \cdots d_0$ *satisfying*

$$n = \sum_{i=0}^{\ell} d_i\, U_i$$

where $d_\ell \neq 0$ *and belonging to the language*

$$\mathcal{L}_{s,m} := \{0, 1, \ldots, s+m-1\}^* \setminus A^*(s+m-1)\{s, s+1, \ldots, s+m-1\}A^*$$

where $A = \{0, 1, \ldots, s+m-1\}$.

The word $d_\ell \cdots d_0$ is said to be the U-representation of n and is denoted by $\text{rep}_U(n)$. For instance, with $s = m = 2$, the first few integers are represented by

$$\varepsilon, 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 100, \ldots, 313, 1000, \ldots$$

Proof. We will use some classical results from the theory of numeration systems. Let β be the dominant root of (6). For the reader familiar with β-expansions, it is clear that $\mathsf{d}_\beta(1) = (s+m-1)s$ and $\mathsf{d}^*_\beta(1) = [(s+m-1)(s-1)]^\omega$. Parry's theorem characterizes β-developments of reals in $[0, 1)$: all their shifts are lexicographically less than $\mathsf{d}^*_\beta(1)$. By Bertrand's theorem from 1989, the language made of greedy U-representations of integers coincides with the language made of the factors occurring in β-developments of reals in $[0, 1)$. See for instance [5, Prop. 2.3.61]. Hence the conclusion follows. $\square$

Analyzing the form of the language $\mathcal{L}_{s,m}$, we see that after a "maximal digit" $s + m - 1$, only digits less than s are allowed. This condition generalizes (4) where only a zero may follow a maximal digit. In particular, $\mathcal{L}_{s,m}$ is a regular language accepted by the DFA on the right in Fig. 1.

Theorem 4.2 (Fraenkel [17, Thm. 5.1]). *A pair* (a, b) *of integers such that* $a \leq b$ *is a* $\mathcal{P}$*-position of the generalized Wythoff's game with parameters* (s, m) *if and only if* $\text{rep}_U(a)$ *ends with an even number of zeroes and* $\text{rep}_U(b)$ *is a left-shift of* $\text{rep}_F(a)$ *where* U *is the numeration system defined by* (6).

With the terminology of [33], if the numeration system is *addable*, then `Walnut` may again be used to prove the above theorem in an automatic way. To that end the language of numeration has to be regular and moreover an *adder* should exist, that is, we need an automaton performing addition.

4.1 Building an adder

Based on the work of Frougny and her coauthors, see [20, 21] and in particular [5, Chap. 2], an adder for these systems can easily be built. We quickly recap its main features. It is a DFA reading triplets of digits and it accepts triplets of valid representations (x, y, z) of the same length (conveniently padded with leading zeros if necessary) such that $\text{val}_U(x) + \text{val}_U(y) = \text{val}_U(z)$.

Let $d := 2(s + m - 1)$. To perform addition, the idea is first to add two numbers digit-wise without carry. So we get a word u over the alphabet $A = \{0, \ldots, d\}$. As an example, with $s = m = 2$, $\text{rep}_U(327) = 12313$ and $\text{rep}_U(1540) = 211302$, addition digit-wise (without carry) amounts to

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 1 & 3 \\ \oplus & 2 & 1 & 1 & 3 & 0 & 2 \\ \hline & 2 & 2 & 3 & 6 & 1 & 5 \end{array}$$

Then compare this word u with any word $w \in A^*$ representing the sum by checking that $\text{val}_U(u) - \text{val}_U(w) = 0$. So subtracting digit-wise u and w, we get a word over the symmetric alphabet $A_d := \{-d, \ldots, 0, \ldots, d\}$ which evaluates to 0. To continue the example, take $w = 231101$ (which is a valid representation) and perform

$$\begin{array}{rrrrrrr} & 2 & 2 & 3 & 6 & 1 & 5 \\ \ominus & 2 & 3 & 1 & 1 & 0 & 1 \\ \hline & 0 & -1 & 2 & 5 & 1 & 4 \end{array}$$

and check that $\mathrm{val}_U((-1)2514) = 0$.

This is the reason to define a DFA, called the *zero automaton* over A_d (see [5, Thm. 2.3.30]). We recall its definition. States are elements in $\mathbb{Z}[\beta]$. Start with the initial state 0. Transitions are of the form

$$s \xrightarrow{\ell} \beta s + \ell, \quad \forall \ell \in A_d.$$

We explore the set of states, starting form 0 using the transitions to discover new states, we only keep states s such that $|s| \leq d/(\beta - 1)$. Starting with 0, we only build new states satisfying the latter condition. Since β is a Pisot number, this DFA is known to be finite (see [5, Thm. 2.3.31]). Final states are of the form $a\beta + b$ such that $aU_1 + bU_0 = 0$, see [21, Lemma 3]. A word $u_k \cdots u_0$ over A_d is accepted by the zero automaton if and only if its U-value is zero, i.e., $\sum_{i=0}^{k} u_i U_i = 0$.

It is easy to modify this automaton to recognize addition: replace the label ℓ of every edge with any triple (a, b, c) such that $a, b, c \in \{0, \ldots s + m - 1\}$ and $a + b - c = \ell$. If u, v, w are three words of the same length (with possibly leading zeroes) then the triple (u, v, w) is accepted by this modified automaton if and only if $\mathrm{val}_U(u) + \mathrm{val}_U(v) = \mathrm{val}_U(w)$. Note that this modified automaton does not check if the words u, v, w are or are not valid representations. This has to be handled separately.

Example 4.3. For $s = t = 2$, the zero automaton has 27 states

$$\Big\{0, -1, 1, -2, 2, \tfrac{5-\sqrt{17}}{2}, \tfrac{\sqrt{17}-5}{2}, 5 - \sqrt{17}, \sqrt{17} - 5, 6 - \sqrt{17}, \sqrt{17} - 6, \tfrac{7-\sqrt{17}}{2}, \tfrac{\sqrt{17}-7}{2},$$
$$4 - \sqrt{17}, \sqrt{17} - 4, \tfrac{3-\sqrt{17}}{2}, \tfrac{\sqrt{17}-3}{2}, 3 - \sqrt{17}, \sqrt{17} - 3, -\tfrac{3(\sqrt{17}-3)}{2}, \tfrac{3(\sqrt{17}-3)}{2},$$
$$2 - \sqrt{17}, \sqrt{17} - 2, \tfrac{1-\sqrt{17}}{2}, \tfrac{\sqrt{17}-1}{2}, \tfrac{11-3\sqrt{17}}{2}, \tfrac{3\sqrt{17}-11}{2}\Big\}$$

If these states are represented by their position in the above ordered list (starting with 1), we get the transition table given in Table 3

$1 \xrightarrow{-2} 4$	$1 \xrightarrow{-1} 2$	$1 \xrightarrow{0} 1$	$1 \xrightarrow{1} 3$	$1 \xrightarrow{2} 5$
$2 \xrightarrow{2} 24$	$2 \xrightarrow{3} 16$	$2 \xrightarrow{4} 6$	$2 \xrightarrow{5} 12$	
$3 \xrightarrow{-5} 13$	$3 \xrightarrow{-4} 7$	$3 \xrightarrow{-3} 17$	$3 \xrightarrow{-2} 25$	
$4 \xrightarrow{5} 22$	$4 \xrightarrow{6} 18$			
$5 \xrightarrow{-6} 19$	$5 \xrightarrow{-5} 23$			
$6 \xrightarrow{-3} 13$	$6 \xrightarrow{-2} 7$	$6 \xrightarrow{-1} 17$	$6 \xrightarrow{0} 25$	
$\vdots$				
$23 \xrightarrow{-6} 25$				
$24 \xrightarrow{4} 24$	$24 \xrightarrow{5} 16$	$24 \xrightarrow{6} 6$		
$25 \xrightarrow{-6} 7$	$25 \xrightarrow{-5} 17$	$25 \xrightarrow{-4} 25$		
$26 \xrightarrow{1} 13$	$26 \xrightarrow{2} 7$	$26 \xrightarrow{3} 17$	$26 \xrightarrow{4} 25$	
$27 \xrightarrow{-4} 24$	$27 \xrightarrow{-3} 16$	$27 \xrightarrow{-2} 6$	$27 \xrightarrow{-1} 12$	

Table 3: Transition table of the zero automaton ($s = m = 2$).

Let us see how one transition is computed. For instance, take state number 26 which is $\frac{11-3\sqrt{17}}{2} = -3\beta + 10$, multiplying by β and adding 1, $-3\beta^2 + 10\beta + 1 = \beta - 5 = (\sqrt{17} - 7)/2$

which is state number 13. Now, from the same state 26, multiplying by β and adding $-\ell$ (with $\ell \geq 0$) gives a negative number $-3\beta^2 + 10\beta - \ell$ whose modulus $\sim 2.438 + \ell$ is larger than $d/(\beta - 1) \sim 2.342$. So there is no transition from state 26 with label $\ell \in \{0, -1, \dots, -6\}$.

The final states are $1, 6, 7, 8, 9$, i.e., the set of final states is

$$\left\{0, \frac{5-\sqrt{17}}{2} = -\beta + 4, \frac{\sqrt{17}-5}{2} = \beta - 4, 5 - \sqrt{17} = -2\beta + 8, \sqrt{17} - 5 = 2\beta - 8\right\}$$

because $U_1 = 4$ and thus $-U_1 + 4U_0 = -2U_1 + 8U_0 = U_1 - 4U_0 = 2U_1 - 8U_0 = 0$.

To finally get the adder over the alphabet $\{0, \dots, s + m - 1\}^3$, replace every label ℓ with all triples (a, b, c) over this alphabet such that $a + b - c = \ell$. (For instance, no transition with label -4, -5 or -6 leads to transitions in the adder because c is less than 4.) Note that these replacements can produce non-accessible states. Then trim the resulting automaton keeping only the states that are both accessible and co-accessible. The resulting DFA has 16 states (4 of them are accepting).

Remark 4.4. In [8], a general procedure to obtain an adder for Dumont–Thomas numeration systems is described and a prototype tool called `licofage` has been implemented. In the setting of this section, the associated morphism is

$$a \mapsto a^{s+t-1}b, \quad b \mapsto a^s.$$

`Walnut` can handle new numeration systems like this one. To do so we provide two automata and store them in the `Custom Bases` directory: an automaton recognizing all valid representations and the adder. We call these `a3ba2` because of the images a^3b and a^2. As mentioned in [8], the validity of the adder can be effectively checked and our adder passes the tests.

```
eval test1 "?msd_a3ba2 Ax,y Ez x+y=z":
eval test2 "?msd_a3ba2 Ax,y,z,t (x+y=z & x+y=t) => z=t":
eval test3 "?msd_a3ba2 Ax,z (x+0=z) <=> x=z":
eval test4 "?msd_a3ba2 Ax,t (x+1=t) <=> (x<t & (Ay x<y => t <= y))":
eval test5 "?msd_a3ba2 Ax,y,z,u,t (u=y+1 & t=z+1)=> (x+y=z <=> x+u=t)":
```

In Table 4, we give the number of states of the adder for various parameters. Note that for

s / m	1	2	3	4	5	6
1	16	17	21	27	33	39
2	12	16	15	21	21	26
3	12	15	16	15	21	21
4	12	13	15	16	15	19
5	12	13	13	15	16	15
6	12	13	13	13	15	16

Table 4: State complexity of the adder for various values of s, m (the sink is not counted).

$s = m = 1$, our procedure gives exactly the Fibonacci adder described in [28] (where the sink was taken into account). An implementation in `Mathematica` providing the necessary `Walnut` file is available online[3].

[3]https://hdl.handle.net/2268/323845

4.2 Back to automatic proofs

The procedure is the same as before. Simply adapt what was done in Section 3. The automaton of Fig. 6 is adapted to a larger alphabet.

```
reg end_even_zeros_32 msd_a3ba2 "0*(00|0*[1-3])*":
def ppos_asym_32 "?msd_a3ba2 $end_even_zeros_32(a) & $left_shift_32(a,b)":
def ppos "?msd_a3b2 $ppos_asym_32(a,b) | $ppos_asym_32(b,a)":
```

Now we define the options (c, d) that are accessible from a given position (a, b). Note that we avoid the use of any (existential) quantifier. Indeed, there exist several equivalent formulas but using quantifiers requires determinization of possibly large automata and cannot be handled. For $m = s = 2$, we define

```
def options_32 "?msd_a3ba2 (a=c & b>d) | (a>c & b=d) |
    (a>c & b>d & ((a+d<=b+c & b+2*c<2*a+d+2)|(b+c<=a+d & a+2*d<2*b+c+2)))":
```

and finally, the two commands below evaluate to `TRUE`

```
eval stable_32 "?msd_a3ba2 Ap,q,r,s (($ppos(p,q) & $ppos(r,s))
   => ~$options_32(p,q,r,s))":
eval absorb_32 "?msd_a3ba2 Ap,q (~$ppos(p,q)
   => Ex,y ($ppos(x,y) & $options_32(p,q,x,y)))":
```

Intermediate computations require respectively a maximum of 730 and 1376 states. Hence we can obtain an automatic proof of Theorem 4.2 for any specific pair (s, m).

5 A new Wythoff-like game in base 2

With `Walnut` at hand, we may experiment a bit and try to define new games. Our aim is to define an invariant game G (i.e., all positions have the same options) whose set of $\mathcal{P}$-positions has a property similar to Theorem 2.5 but this time expressed in base 2 (and not in the Fibonacci system), i.e., a pair (a, b) of integers such that $a \leq b$ is a $\mathcal{P}$-position of G if and only if the base-2 representation $\text{rep}_2(a)$ ends with an even number of zeroes and $\text{rep}_2(b)$ is a left-shift of $\text{rep}_2(a)$, i.e., $b = 2a$. So, the first few $\mathcal{P}$-positions are

$$(A_n, B_n)_{n\geq 0} = (0, 0), (1, 2), (3, 6), (4, 8), (5, 10), (7, 14), \ldots.$$

In this case, we have a partition of $\mathbb{N}$ as the union of the A_n's and B_n's. Note that the sequence (A_n) appears in the OEIS as A003159. We can thus proceed exactly as in the previous section:

```
reg g_end_even_zeros msd_2 "0*(00|0*1)*":
reg left_shift {0,1} {0,1} "([0,0]|([0,1][1,1]*[1,0]))*":
def g_ppos_asym "$g_end_even_zeros(a) & $left_shift(a,b)":
def g_ppos "$g_ppos_asym(a,b) | $g_ppos_asym(b,a)":
```

We remark that if the numeration base is not stated via the `?msd_` syntax, then `Walnut` uses base 2 representations. Now we define the binary predicate `g_move` for the maximal set of moves as the set of all pairs not in (1)

```
def g_forbidden "Ep,q,s,t ($g_ppos(p,q) & $g_ppos(s,t)
                                    & p<=s & q<=t & a+p=s & b+q=t)":
def g_move "~$g_forbidden(a,b)":
```

The set of moves written in base 2 is described by the DFA depicted in Fig. 8. In some sense, one can think that this is a "simple" game since the set of moves (written in base-2) is accepted by a DFA. The automaton could thus be distributed at the beginning of the game to the players since it fully describes the rules of the game. Note that the set of moves is symmetric and this can be readily verified

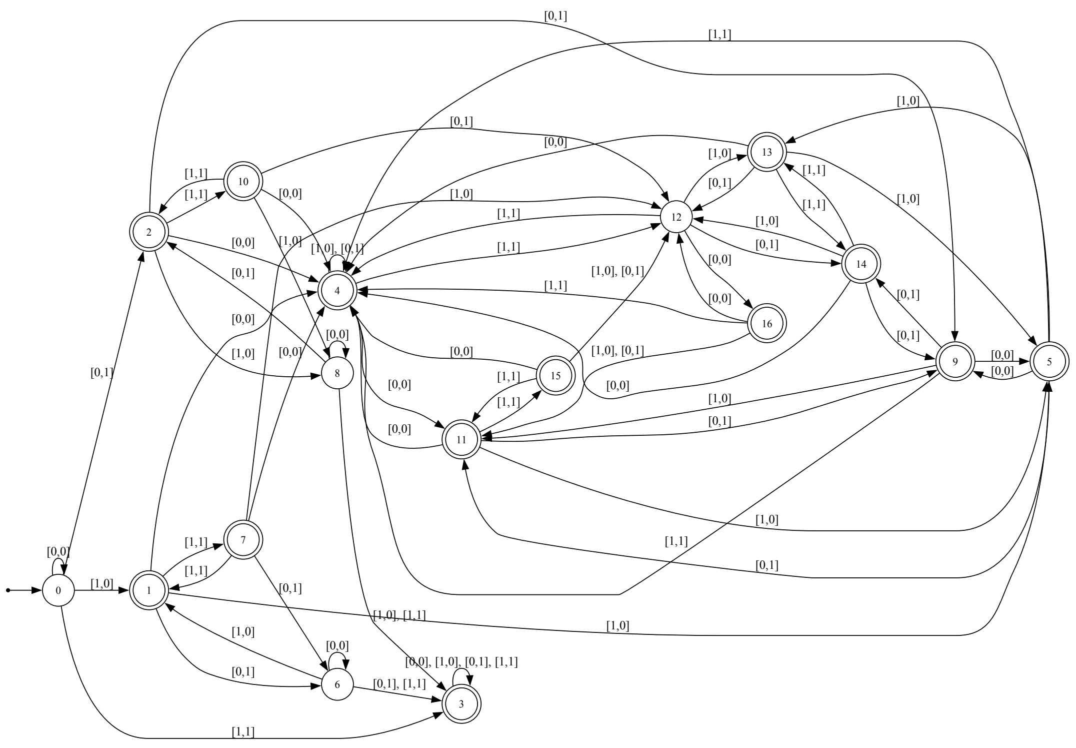
Figure 8: The DFA recognizing moves in base 2 of the game whose $\mathcal{P}$-positions have the form $\text{rep}_2(A_n) \in (0^*1 + 00)^*$ and $B_n = 2A_n$.

```
eval test_symmetry "Ax,y $g_move(x,y)=>$g_move(y,x)":
```

We have to check that with this rule-set, the set of $\mathcal{P}$-positions is absorbing. The following command evaluates to TRUE.

```
eval g_absorbing "Ap,q (~$g_ppos(p,q) => Ex,y (x<=p & y<=q &
                              $g_ppos(x,y) & $g_move(p-x,q-y)))":
```

By construction, the set of $\mathcal{P}$-positions is stable. But this is just a one-line code in Walnut (which of course evaluates to TRUE).

```
eval test_stable "Aa,b ( ($g_ppos(a,b) & ~(a=0&b=0))
    => ~(Ec,d ($g_ppos(c,d) & a<=c & c<=d & $g_move(c-a,d-b))))":
```

Remark 5.1. If we change just a bit the aimed set of $\mathcal{P}$-positions. For instance, by adding just one extra $\mathcal{P}$-position $(1,1)$ as below.

```
def g2_ppos_asym "($g_end_even_zeros(a) & $left_shift(a,b))|(a=1&b=1)":
```

Then proceeding exactly as above does not provide an absorbing set of expected $\mathcal{P}$-positions.

The $\mathcal{P}$-positions of Wythoff's game are *coded* by the infinite Fibonacci word $abaababaab\cdots$ because $A_n = \lfloor n\varphi \rfloor$ where φ is the golden ratio. See, for instance, see [10]. For many combinatorial games, it is desirable to have a similar property. Having an infinite word where the index of the nth occurrence of a (resp. b) is A_n (resp. B_n). If the game is played on k heaps, then we have a k-letter alphabet. Such an infinite word always exists but we are looking for words having a combinatorial structure. In the base-2 game, we want a 2-automatic sequence coding the $\mathcal{P}$-positions. We inspect the automaton in Fig. 9 resulting from the evaluation of the command

```
def g_An "Ex $g_ppos_asym(a,x)":
```

This automaton accepts the base-2 expansions of the A_n's. With this DFA is associated the

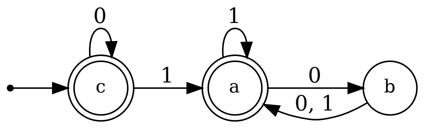

Figure 9: A DFA recognizing $\text{rep}_2(A_n)$ of the modified game.

2-uniform morphism

$$f : \begin{cases} c \to ca \\ a \to ba \\ b \to aa \end{cases}$$

generating the infinite word

$$f^\omega(c) = cabaaabababaaabaaabaaabababaaaba\cdots$$

The positions (counted from 0) of the nth letters a and b provide the nth $\mathcal{P}$-position (A_n, B_n) in this game. Note that this word is nothing else than a shift of the period-doubling word, the fixed point of the morphism $0 \mapsto 01$, $1 \mapsto 00$ (see, for instance, [1, Ex. 6.3.4]). This can easily be proved in Walnut;

```
morphism pd "0->01 1->00";
morphism shiftpd "0->01 1->21 2->11";
promote X pd;
promote Y pd;
eval is_shift "An (X[n]=@0 <=> Y[n+1]=@1) & (X[n]=@1 <=> Y[n+1]=@2)";
```

and the last command returns TRUE.

Remark 5.2. The rule-set contains all the moves from the original Wythoff's game because the following commands evaluate to TRUE

```
eval test "Aa (a>0 => $g_move(a,0))":
eval test "Aa (a>0 => $g_move(a,a))":
```

A real human player may complain that the rule-set is given by an intricate automaton. If we want to determine the allowed moves of the form $(1, n)$, we use

```
eval test n "$g_move(1,n)":
```

to get a Maple file containing the relevant linear representation. From this, we find that for $1 \le n \le 50$, all moves $(1, n)$ are allowed except for $n = 2, 5, 11, 14, 17, 23, 29, 35, 38, 41, 47, 50$. We can therefore check that the move $(1, 4)$ allowed in this game is not a valid Wythoff-move because

```
eval test "?msd_fib $wythoff_forbidden(1,4)":
```

returns TRUE.

Remark 5.3. We can also change the status of the states 7, 10, 15, 16 in Fig. 8 to non-accepting (so we have a smaller set of moves) and still check with such an alternative predicate for g_move that the set of $\mathcal{P}$-positions is still absorbing (stability is not affected when removing moves).

5.1 Is this really a new game?

Having found (A_n) in the OEIS, we have discovered that a game with this set of $\mathcal{P}$-positions has been considered by Fraenkel in [18]. The author considers a game G_2 with the moves of Nim but also the possibility to remove a positive number of tokens from each pile, say k and ℓ, so that $|k-\ell|$ isn't too large with respect to the position (x_1, y_1) moved to from (x_0, y_0), namely, $|k-\ell| < x_0 - x_1$ $(x_1 \leq y_1)$. With [18, Thm. 2], he proved that for this game the set of $\mathcal{P}$-positions is indeed made of the (A_n, B_n). Since the set of $\mathcal{P}$-positions is 2-automatic and the options can also be defined in Presburger arithmetic, then we can again produce an automatic proof of this result.

One of the referees of this paper asked the following question. Let $\alpha \geq 3$ be an integer. Does there exist a combinatorial game whose $\mathcal{P}$-positions are exactly of the form $(n, \alpha n)$?

6 Conclusion

On the one hand, numeration systems provide a framework for representing numbers using integer bases or more general expansions. They play a crucial role in the analysis of combinatorial game strategies. For instance, Nim-sum (or digit-wise XOR) is performed on binary representations of positions and permits to characterize the $\mathcal{P}$-positions of Nim game. As it has been widely used in Fraenkel's pioneering work [16, 17, 18], numeration systems allows one for efficient encoding of game positions and strategies, facilitating the understanding of winning conditions.

On the other hand, numeration systems and sets of integers recognized by finite automata are the core of the Büchi–Bruyère theorem. With this paper, we fill the gap and show that `Walnut` is perfectly suited to the context of combinatorial games obtaining a variety of results. Some are short automatic proofs of already known results. Their derivations are based on the fact that the set of $\mathcal{P}$-positions is, in fact, recognizable by a finite automaton once represented in an appropriate numeration system see Theorems 2.5, 2.15 and 4.2. Equivalently, the infinite word coding the $\mathcal{P}$-positions is morphic (to make the connection with combinatorics on words as briefly discussed in the last section). This fact is exemplified by considering several variations of Wythoff's game that are intimely related to the Fibonacci numeration system or some of its generalizations. In particular, decidability depends on the fact that addition is recognized by finite automaton. We show how to effectively get this automaton. It would be interesting to study the structure of the obtained adder automata and their state-complexity, see Table 4. We easily obtained a series of new results, including the (automatic) proof of a more than 15-year old conjecture [11]. `Walnut` provides a great tool for combinatorial exploration as shown by our study of redundant moves for a variation of Wythoff's game. It can also be used for Grundy values as soon as the corresponding domain is represented by a regular language [24].

Our contribution also highlights the limitations of the approach. Regardless of the available computing resources, not all properties are easily expressible by first-order logic. This is especially true when one aims to address a question for a family of games that depend on one or more parameters.

Finally, other variations and generalizations of Wythoff's game have been considered in the literature. For instance, in [23, 25, 19], the rules of the considered game could be defined in Presburger arithmetic. This is a first step that needs to be validated in order to implement the formalism developed in this article. However, it is not obvious to obtain a numeration system with good properties in which the $\mathcal{P}$-positions can be recognized by a finite automaton. Our paper therefore opens up to new questions for investigation.

Acknowledgments

We warmly thank Jeffrey Shallit for useful tips and comments about `Walnut`.

Appendix. Supplementary material

Source files are available at this address: `https://hdl.handle.net/2268/323845`

References

[1] Jean-Paul Allouche and Jeffrey Shallit. *Automatic sequences: Theory, applications, generalizations*. Cambridge University Press, Cambridge, 2003.

[2] Aseem Baranwal, Luke Schaeffer, and Jeffrey Shallit. Ostrowski-automatic sequences: theory and applications. *Theoret. Comput. Sci.*, 858:122–142, 2021. `doi:10.1016/j.tcs.2021.01.018`.

[3] Claude Berge. *The theory of graphs. Translated from the 1958 French edition by Alison Doig.* Mineola, NY: Dover Publications, 2nd printing of the 1962 first English edition edition, 2001.

[4] Valérie Berthé. Autour du système de numération d'Ostrowski. *Bull. Belg. Math. Soc. Simon Stevin*, 8(2):209–239, 2001. URL: `http://projecteuclid.org/euclid.bbms/1102714170`.

[5] Valérie Berthé and Michel Rigo, editors. *Combinatorics, automata and number theory*, volume 135 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2010. `doi:10.1017/CBO9780511777653`.

[6] Véronique Bruyère, Georges Hansel, Christian Michaux, and Roger Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc. - Simon Stevin*, 1(2):191–238, 1994.

[7] Arturo Carpi and Cristiano Maggi. On synchronized sequences and their separators. *Theor. Inform. Appl.*, 35(6):513–524 (2002), 2001. A tribute to Aldo de Luca. `doi:10.1051/ita:2001129`.

[8] Olivier Carton, Jean-Michel Couvreur, Martin Delacourt, and Nicolas Ollinger. Addition in Dumont-Thomas numeration systems in theory and practice, 2024. URL: `https://arxiv.org/abs/2406.09868`, `arXiv:2406.09868`.

[9] Émilie Charlier, Narad Rampersad, and Jeffrey Shallit. Enumeration and decidable properties of automatic sequences. *Int. J. Found. Comput. Sci.*, 23(5):1035–1066, 2012. `doi:10.1142/S0129054112400448`.

[10] Eric Duchêne, Aviezri S. Fraenkel, Vladimir Gurvich, Nhan Bao Ho, Clark Kimberling, and Urban Larsson. Wythoff visions. In *Games of no chance 5*, volume 70 of *Math. Sci. Res. Inst. Publ.*, pages 35–87. Cambridge Univ. Press, Cambridge, 2019.

[11] Eric Duchêne, Aviezri S. Fraenkel, Richard J. Nowakowski, and Michel Rigo. Extensions and restrictions of Wythoff's game preserving its $\mathcal{P}$ positions. *J. Comb. Theory, Ser. A*, 117(5):545–567, 2010. `doi:10.1016/j.jcta.2009.07.010`.

[12] Eric Duchêne and Michel Rigo. Invariant games. *Theoret. Comput. Sci.*, 411(34-36):3169–3180, 2010. `doi:10.1016/j.tcs.2010.05.007`.

[13] R. Fokkink. Wie weet wie Willem Wijthoff was? *Nieuw Archief voor Wiskunde*, 17(7):275–280, 2016.

[14] Robbert Fokkink, Gerard Francis Ortega, and Dan Rust. Corner the empress, 2022. URL: `https://arxiv.org/abs/2204.11805`, `arXiv:2204.11805`.

[15] Robbert Fokkink and Dan Rust. Queen reflections: a modification of Wythoff Nim. *International Journal of Game Theory*, pages 1432–1270, 2022. `doi:10.1007/s00182-022-00824-1`.

[16] Aviezri S. Fraenkel. How to beat your Wythoff games' opponent on three fronts. *Am. Math. Mon.*, 89:353–361, 1982. `doi:10.2307/2321643`.

[17] Aviezri S. Fraenkel. Heap games, numeration systems and sequences. *Ann. Comb.*, 2(3):197–210, 1998. doi:10.1007/BF01608532.

[18] Aviezri S. Fraenkel. New games related to old and new sequences. *Integers*, 4:G6, 18, 2004.

[19] Eric Friedman, Scott M. Garrabrant, Ilona K. Phipps-Morgan, A. S. Landsberg, and Urban Larsson. Geometric analysis of a generalized Wythoff game. In *Games of no chance 5*, volume 70 of *Math. Sci. Res. Inst. Publ.*, pages 343–372. Cambridge Univ. Press, Cambridge, 2019.

[20] Christiane Frougny. Representations of numbers and finite automata. *Math. Systems Theory*, 25(1):37–60, 1992. doi:10.1007/BF01368783.

[21] Christiane Frougny and Boris Solomyak. On representation of integers in linear numeration systems. In *Ergodic theory of $\mathbf{Z}^d$ actions (Warwick, 1993–1994)*, volume 228 of *London Math. Soc. Lecture Note Ser.*, pages 345–368. Cambridge Univ. Press, Cambridge, 1996. doi:10.1017/CBO9780511662812.014.

[22] Nhan Bao Ho. Two variants of Wythoff's game preserving its $\mathcal{P}$-positions. *J. Comb. Theory, Ser. A*, 119(6):1302–1314, 2012. doi:10.1016/j.jcta.2012.03.010.

[23] Urban Larsson. A generalized diagonal Wythoff Nim. *Integers*, 12(5):1003–1027, 2012. doi:10.1515/integers-2012-0020.

[24] Urban Larsson and Nathan Fox. An aperiodic subtraction game of nim-dimension two. *J. Integer Seq.*, 18(7):Article 15.7.4, 6, 2015.

[25] Urban Larsson and Johan Wästlund. Maharaja Nim: Wythoff's queen meets the knight. *Integers*, 14:Paper No. G05, 21, 2014.

[26] Arnaud Maes. An automata-theoretic decidability proof for first-order theory of $\langle \mathbf{N}, <, \mathrm{P} \rangle$ with morphic predicate P. *J. Autom. Lang. Comb.*, 4(3):229–245, 1999. Journées Montoises d'Informatique Théorique (Mons, 1998).

[27] Hamoon Mousavi. Automatic theorem proving in Walnut, 2016. arXiv:1603.06017.

[28] Hamoon Mousavi, Luke Schaeffer, and Jeffrey Shallit. Decision algorithms for Fibonacci-automatic words, I: Basic results. *RAIRO Theor. Inform. Appl.*, 50(1):39–66, 2016. doi:10.1051/ita/2016010.

[29] Michel Rigo. *Formal languages, automata and numeration systems. Vol. 2. Applications to recognizability and decidability.* Hoboken, NJ: John Wiley & Sons; London: ISTE, 2014. doi:10.1002/9781119042853.

[30] Michel Rigo. From combinatorial games to shape-symmetric morphisms. In *Substitution and tiling dynamics: introduction to self-inducing structures. Lecture notes from the research school on tiling dynamical systems, CIRM Jean-Morlet Chair, Marseille, France, Fall 2017*, pages 227–291. Cham: Springer, 2020. doi:10.1007/978-3-030-57666-0_5.

[31] Jeffrey Shallit. Synchronized sequences. In *Combinatorics on words*, volume 12847 of *Lecture Notes in Comput. Sci.*, pages 1–19. Springer, Cham, [2021] ©2021. URL: https://doi.org/10.1007/978-3-030-85088-3_1, doi:10.1007/978-3-030-85088-3_1.

[32] Jeffrey Shallit. Using Automata and a Decision Procedure to Prove Results in Pattern Matching. In Hideo Bannai and Jan Holub, editors, *33rd Annual Symposium on Combinatorial Pattern Matching (CPM 2022)*, volume 223 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:3, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16129, doi:10.4230/LIPIcs.CPM.2022.2.

[33] Jeffrey Shallit. *The logical approach to automatic sequences. Exploring combinatorics on words with Walnut*, volume 482 of *Lond. Math. Soc. Lect. Note Ser.* Cambridge: Cambridge University Press, 2023. `doi:10.1017/9781108775267`.

[34] Jeffrey Shallit. Some Tribonacci conjectures. *Fibonacci Quart.*, 61(3):214–221, 2023.

[35] Jeffrey Shallit. Proof of Irvine's conjecture via mechanized guessing. *Integers*, 24:Paper No. A51, 17, 2024.

[36] W. A. Wythoff. A modification of the game of Nim. *Nieuw Arch. Wiskd., II. Ser.*, 7:199–202, 1906.

[37] E. Zeckendorf. Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas. *Bull. Soc. Roy. Sci. Liège*, 41:179–182, 1972.