A Combinatorial Game whose Solution Gives an Efficiently Computable Shift-Rule for the Reversed Prefer-One De Bruijn Sequence

Gera Weiss
Department of Computer Science
Ben Gurion University of the Negev
Israel, Gera Weiss[a]

[a]*Department of Computer Science, Ben-Gurion University of The Negev*

Abstract

We define a simple combinatorial game called the shift game. The game is played by two player, Alice and Bob, over a binary shift register. It goes by shifting the bits in the register to the right where the right-most bit drops out and the players choose which bit to insert from the left. If the bit being dropped out is one, Bob chooses the bit to insert, otherwise Alice does. Alice's goal is to have zeroes in all the bits of the register and Bob goal is to block her from achieving this. An online version of the game where the computer plays Bob's role and the user plays Alice's role is available at `www.cs.bgu.ac.il/~geraw/a-combinatorial-game.html`. We analyze this game and give an efficiently computable optimal strategies for both Bob and Alice. We show that if both players play optimally and the initial state of the register is $\langle 0, \ldots, 0 \rangle$, the resulting play is the reverse of the well-known prefer-one De Bruijn sequence. This gives an efficiently computable shift-rule for this sequence. We also show that similar relations exists between any given De Bruijn sequence and a game that we define for it.

Keywords: De Bruijn sequence, Ford sequence, Combinatorial Games, prefer-one sequence, shift rule
2010 MSC: 94A55, 05C45, 05C38

Email address: `geraw@cs.bgu.ac.il` (Gera Weiss)

1. Introduction

A binary De Bruijn sequence of order n is a sequence of bits (elements of $\{0, 1\}$) that contains every possible substring of length n exactly once. There are $2^{2^{n-1}-n}$ De Bruijn sequences for each order n (see [3] for a trace of the origins of this result). One such sequence, which is the focus of this paper, can be constructed using the prefer-one algorithm [5]: Given an integer $n \geq 1$ as a parameter, the algorithm puts n zeroes, then it repeatedly tries 1 as the next bit and commit to it when the word formed by the last n bits has not been encountered previously in the sequence, otherwise the 1 is rolled back and 0 is added. The process stops when both 0 and 1 do not introduce a new word. In this paper, for notational convenience, we move the n leading zeroes to the end of the sequence.

A shift-rule for a De Bruijn sequence of order n is a function that maps each substring of length n to the next substring of length n in the sequence [6]. In this paper we use combinatorial game theory [2] to develop an efficiently computable ($O(n)$ time and memory) shift-rule for the reversed prefer-one sequence. Beyond the formulation of the rule, the paper reveals a new connection between the combinatorics of De Bruijn sequences and games. In addition to the game that we analyze in full details, whose solution gives the shift-rule for the reversed prefer-one sequence, we propose a way to transform any De Bruijn sequence to a game that corresponds to the sequence in the same way.

The paper proceeds as follows. In Section 2 we define the combinatorial game. In Section 3 we give an inductive proof that Alice can always win in this game. In Section 4 we show that a solution to the game gives a shift-rule for the reversed prefer-one sequence. In Section 5 we provide efficiently computable strategies for both Alice and Bob. In Section 6 we translate these results to an efficient shift-rule for the reversed prefer-one sequence. In Section 7 we propose a family of games that corresponds to all the De Bruijn sequences.

2. A combinatorial game

Definition 1. The n-bit *shift game* is played by two players, Alice and Bob, as follows. The initial state of the game is the n-tuple $s_0 = \langle 0, \ldots, 0 \rangle$ and a play of the game consists of the sequence $\langle s_t \rangle_{t=0}^{\infty}$, such that if $s_t =$

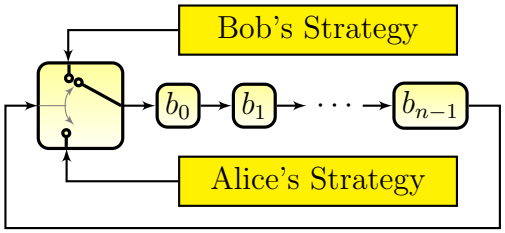


Figure 1: The shift game as a shift register. The rightmost bit, b_{n-1}, determines whether Alice's strategy or Bob's strategy will set the next bit to enter the shift-register at b_0 (which, in turn, will push the content of the register rightward).

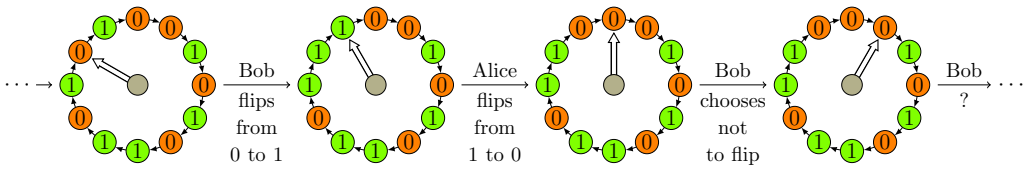
$\langle b_0, \ldots, b_{n-1} \rangle$ then

$$s_{t+1} = \begin{cases} \langle A(s_0, \ldots, s_t), b_0, \ldots, b_{n-2} \rangle & \text{if } b_{n-1} = 1; \\ \langle B(s_0, \ldots, s_t), b_0, \ldots, b_{n-2} \rangle & \text{if } b_{n-1} = 0; \end{cases}$$

where $A\colon (\{0,1\}^n)^* \to \{0,1\}$ is Alice's strategy and $B\colon (\{0,1\}^n)^* \to \{0,1\}$ is Bob's strategy. Alice wins the game if there exists $t > 0$ such that $s_t = s_0$ and Bob wins otherwise.

This game was first introduced in [7] for the purpose of solving a problem in control theory. Here, we show how the game is related to De Bruijn sequences, solve it, and use the solution to devise an efficiently computable shift-rule for a De Bruijn sequence.

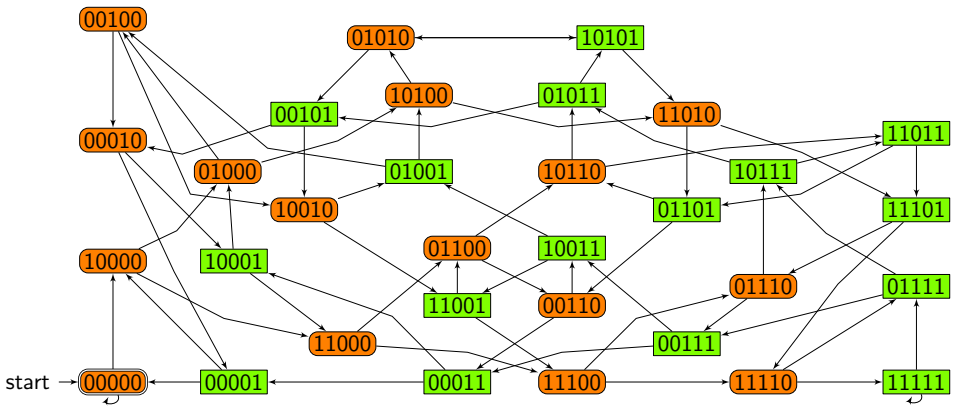
A direct visualization of the game is given in Figure 1. Another way to visualize the shift game is to imagine n two-state switches at the periphery of a dial and a clock-like hand that moves cyclically and, at each game turn, points at one of them:



If the switch that the hand points at is at a 1 state, Alice can flip it to 0 or leave it at 1. If the switch is 0, its Bob's choice of whether to flip it to 1 or leave it as it is. Then, the hand moves to the next switch, say clockwise, and the game repeats. The game begins with all the switches

at 0. Alice wins if the game comes to this state again, and Bob wins otherwise. We call the switch that the handle points at in the beginning of the game sw_0, the switch that is next to it clockwise sw_1, and so on up to sw_{n-1}. If we denote the state of the ith switch at turn number t by $sw_i(t) \in \{0,1\}$ and the position of the hand by $h(t) = t \bmod n$, it is easy to see that the relation between this representation of the game and the one given in Definition 1 is $s_t = \langle sw_{h(t)-1}(t), \ldots, sw_{h(t)-n}(t)\rangle$ where the arithmetic for switch indexes is modulo n (a negative index, $-i$, is considered as $n-i$). For example, the states of the game depicted above are: $\cdots \to \langle 1,0,1,1,0,1,0,1,0,0,1,0\rangle \to \langle 1,1,0,1,1,0,1,0,1,0,0,1\rangle \to \langle 0,1,1,0,1,1,0,1,0,1,0,0\rangle \to \langle 0,0,1,1,0,1,1,0,1,0,1,0\rangle \to \cdots$.

Another way to visualize this game is as a Büchi game over a De Bruijn graph, as follows. A Büchi game is a two-player combinatorial game played by Alice and Bob over a finite directed graph with a designation of a set of initial vertices and a set of accepting vertices whose vertices are partitioned into squares and ellipses. A game play goes as follows: A token is positioned on an (arbitrary) initial vertex. At each game round, the token is moved to a successor vertex. If the token is on a square Alice choses a successor and places moves to it. If the token is on an ellipse, Bob chooses the successor and moves the token to it. Alice wins if the token visits an accepting vertex infinitely often and Bob wins otherwise.



The shift game can be viewed as a Büchi game played over the n-dimensional binary De Bruijn graph, depicted above for $n = 5$, as follows. The vertices of the graph are all possible binary sequences of length n. If one of the vertices can be expressed as another vertex by shifting all its bits by one place to the right and adding a new bit at the end of this vertex, then

the latter has a directed edge to the former. The vertices whose rightmost bit is zero are ellipses and the vertices whose rightmost bit is one are squares. The only initial state is 0^n and it is also the only accepting state. Clearly, the game given in Definition 1 is equivalent to the Büchi game over the n-dimensional binary De Bruijn graph.

3. Alice can always win

Our first result is that Alice can always win the shift game, for any n:

Proposition 2. *There is a strategy for Alice that wins the n-bit shift game against any strategy that Bob may use.*

Proof. To allow a proof by induction, we will prove the stronger claim that Alice can win even a more challenging game where the game begin in any arbitrary state (that, say, Bob can choose) and where she wins only if there are infinitely many $k \in \mathbb{N}$ such that $s_{k \cdot n} = \langle 0, \ldots, 0 \rangle$. In other words, not only that Alice has to force the state of the game to be $\langle 0, \ldots, 0 \rangle$, she has to arrive at this state after a number of turns that is a multiple of n and she has to do it repeatedly.

The proof goes by induction on n. If $n = 1$, Alice has a trivial winning strategy: If the initial state is $\langle 1 \rangle$ Alice can chose the second state to be $\langle 0 \rangle$ and win. If the initial state is $\langle 0 \rangle$ Bob can either choose the second state to be $\langle 0 \rangle$ and then Alice wins, or have the second state be $\langle 1 \rangle$ and then Alice can choose the third state to be $\langle 0 \rangle$ and win. Since Alice can do this repeatedly, this proves also the stronger claim.

For the induction step, we will use the visualization of the game as n two-state switches at the periphery of a dial. Assume, by induction, that Alice has a winning strategy for n switches. The proof goes by providing a winning strategy for Alice in the game over $n + 1$ switches in which the assumed strategy (over n switches) is used as a black box.

The trick for Alice is to ignore the first switch (sw_0) and pretend that she is playing the game only over the other switches. When the hand leaves the last switch (sw_{n-1}) and moves on, Alice pretends that it did not stop at the first switch as if it moved directly to the second one, as if the first switch does not exist. If Alice has to make a decision about the first switch, when it is 1 and the hand points at it, she always leaves it as it is.

Now, by the induction hypothesis, as Alice pretends that she is playing a game over n switches, she can make them all 0. Moreover, as she can win the

more challenging game described at the beginning of the proof, she can do so after a number of turns that divides n, i.e, she can force arrival to a state where all the switches but the first are 0 and the hand points at the first switch. At this game turn, the first switch may be 1 or 0. If it is 1, Alice can turn it to 0 and win the game over $n+1$ switches. If it is 0, Bob may choose to leave it 0 or switch it to 1. If Bob leaves the switch 0, Alice again wins. If, on the other hand, Bob chooses to turn the switch to 1, Alice can continue playing her strategy and end up, once more, with all the switches but the first at 0 and the hand pointing at the first switch. This time, however, the first switch is at the 1 position because Alice leaves it as it is whenever the hand passes it. Thus, also in this case, Alice wins the game over $n+1$ bits. □

Note that the strategy proposed in the proof is not memoryless, i.e., Alice's move depends not only on the state of the game but also on the position of the handle, i.e., on the turn number modulo n. Formally, a strategy $A\colon (\{0,1\}^n)^* \to \{0,1\}$ is called memoryless if there is a function $\hat{A}\colon \{0,1\}^n \to \{0,1\}$ such that $A(s_1,\ldots,s_n) = \hat{A}(s_n)$. To shorten notation, we will just say, from now on, that $\hat{A}$ is a memoryless strategy for Alice. The following corollary establishes that there is a winning memoryless strategy for Alice:

Corrolary 3. There exists a memoryless strategy for Alice that wins against any (not necessarily memoryless) strategy that Bob may play.

Proof. Since the rules of the game are memoryless in the sense that at any time the remainder of the game depends only on the current state, not on the history that lead to this state, there is no need for Alice to remember anything about the history. □

4. The optimal strategies correspond to a shift-rule for the reversed prefer-one sequence

Now, that we know that Alice can always win the game, the next question is how many turns must pass, if Bob plays well, until the state $\langle 0,\ldots,0\rangle$ is reached. For this discussion, we imagine that, at the end of the game, Alice pays Bob an amount proportional to the number of turns that passed until she wins. Thus, Alice's objective is now to reach the state $\langle 0,\ldots,0\rangle$ as quick as possible and Bob's objective is to delay the inevitable arrival to that state as much as he can.

The arguments we used in the proof of Corollary 3 give us that there is an optimal strategy for Alice that is memoryless and that there is an optimal strategy for Bob that is memoryless. This means that Alice can force, using a memoryless strategy, that no state is repeated before the state $\langle 0, \ldots, 0 \rangle$ is reached. By the pigeonhole principle, we get that Bob cannot force a delay of more than 2^n turns before reaching $\langle 0, \ldots, 0 \rangle$. We will now introduce tools to show that Bob has a memoryless strategy that forces a delay of at least 2^n turns, no matter how Alice plays. Since this is the best Bob can expect, we call this strategy optimal for Bob.

Definition 4 (Martin [5])**.** The reversed prefer-one sequence of order n, $\langle s_t \rangle_{t=0}^{2^n-1}$, is defined inductively from its end to its beginning such that $s_{2^n-1} = \langle 0, \ldots, 0, 1 \rangle$, and if $s_t = \langle b_0, \ldots, b_{n-1} \rangle$ then

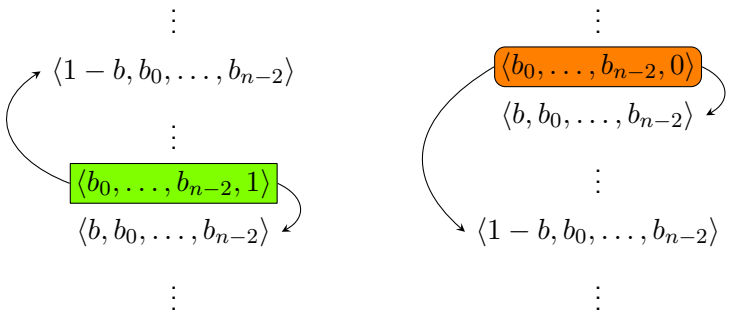
$$s_{t-1} = \begin{cases} \langle b_1, \ldots, b_{n-1}, 1 \rangle & \langle b_1, \ldots, b_{n-1}, 1 \rangle \notin \{s_i\}_{i=t}^{2^n-1}; \\ \langle b_1, \ldots, b_{n-1}, 0 \rangle & \text{otherwise.} \end{cases}$$

Theorem 5 (Martin [5])**.** *The sequence $\langle s_t \rangle_{t=0}^{2^n-1}$, given in Definition 4, is a permutation of the set of all n-ary binary tuples, namely $\{s_t\}_{t=0}^{2^n-1} = \{0,1\}^n$.*

In the following two claims we show that the sequence specified in Definition 4 is such that: (1) if we look at a state from which Bob plays (the rightmost bit is 0) the options that Bob have are to step to the next state in the sequence or to jump to a state that appears later in the sequence; (2) if we look at a state from which Alice plays (the rightmost bit is 1) the options that Alice have are to step to the next state in the sequence or to jump to a state that appears earlier in the sequence. This is illustrated in Figure 2. We will later show (in propositions 8 and 9) that this means that the only optimal strategies for Bob and for Alice are to follow this sequence.

Claim 6. *In the sequence $\langle s_t \rangle_{t=0}^{2^n-1}$ given in Definition 4, if $s_t = \langle b_0, \ldots, b_{n-2}, 0 \rangle \neq 0^n$ and $s_{t+1} = \langle b, b_0, \ldots, b_{n-2} \rangle$ then the (unique) index t' such that $s_{t'} = \langle 1-b, b_0, \ldots, b_{n-2} \rangle$ must be bigger than $t+1$.*

Proof. By the definition of the sequence we have $s_{t'-1} \in \{\langle b_0, \ldots, b_{n-2}, 0 \rangle, \langle b_0, \ldots, b_{n-2}, 1 \rangle\}$. Because $s_{t+1} \neq s_{t'}$, we get that $s_t \neq s_{t'-1}$ which leaves us only the option $s_{t'-1} = \langle b_0, \ldots, b_{n-2}, 1 \rangle$. As $s_{t'-1}$ and s_t differ only in the last bit and the last bit of $s_{t'-1}$ is one, the "prefer-one" construction will put it closer to the end of the sequence, i.e., $t' - 1 > t$, which implies that t' is bigger than $t+1$. □



(a) When its Alice's turn, her options are the next state on the sequence or a state that appears earlier in the sequence.

(b) When its Bob's turn, his options are the next state on the sequence or a state that appears later in the sequence.

Figure 2: A graphical representation of the results in claims 6 and 7.

Claim 7. *In the sequence* $\langle s_t\rangle_{t=0}^{2^n-1}$ *given in Definition 4, if* $s_t = \langle b_0, \ldots, b_{n-2}, 1\rangle$ *and* $s_{t+1} = \langle b, b_0, \ldots, b_{n-2}\rangle$ *then the (unique) index* t' *such that* $s_{t'} = \langle 1-b, b_0, \ldots, b_{n-2}\rangle$ *must be smaller than* $t+1$.

Proof. As in the proof of Claim 6, we have $s_{t'-1} \in \{\langle b_0, \ldots, b_{n-2}, 0\rangle, \langle b_0, \ldots, b_{n-2}, 1\rangle\}$ and $s_{t+1} \neq s_{t'}$ implies that $s_t \neq s_{t'-1}$ which leaves us only with the option $s_{t'-1} = \langle b_0, \ldots, b_{n-2}, 0\rangle$. Again, $s_{t'-1}$ and s_t differ only in the last bit and, because now the last bit of $s_{t'-1}$ is zero, the "prefer-one" construction will put it closer to the beginning of the sequence, i.e., $t'-1 < t$, which implies that t' is smaller or equal to t. □

For the following two propositions, consider the memoryless strategy $S\colon \{0,1\}^n \to \{0,1\}$ that maps each tuple s_t in the sequence $\langle s_t\rangle_{t=0}^{2^n-1}$, given in Definition 4, to the leftmost bit of s_{t+1} or, if $t = 2^n - 1$, to zero.

Proposition 8. *The only memoryless strategy for Alice that wins the game against any strategy that Bob may use is* S.

Proof. The fact that if both players apply the strategy S then Alice wins is clear because the sequence $\langle s_t\rangle_{t=0}^{2^n-1}$ ends with $\langle 0, \ldots, 0, 1\rangle$ and at this state Alice plays 0 and wins. Assume, towards contradiction, that Bob plays S and there is a game play $\hat{s}_0, \ldots, \hat{s}_m, 0^n$ where Alice wins at the mth turn without following the strategy S. Let m be the minimal such index. Let

$k \leq m$ be the first index such that the left-most bit of $\hat{s}_{k+1}$ is not $S(\hat{s}_k)$. Since Bob applies the strategy S, it must be Alice that did not follow S, so the right-most bit of $\hat{s}_k$ must be one. From the minimality of k, by the definition of S, we get that $\hat{s}_0 = s_0, \ldots, \hat{s}_{k-1} = s_{k-1}$ where $s_0, \ldots, s_{k-1}$ is a prefix of the sequence given in Definition 4. Specifically, by Claim 7, we get that $\hat{s}_{k+1} \in \{\hat{s_0}, \ldots, \hat{s}_k\}$ which means that a state is repeated. This is in contradiction with the assumption that Alice won the game at the mth state because, when both players apply memoryless strategies, a repetition of a state means that the game is going to loop forever. The repeated part cannot contain the state 0^n because of the minimality of m. □

Proposition 9. *If Bob plays the strategy S he forces that the game goes for at least 2^n rounds before Alice's win.*

Proof. We can assume without loss of generality that Alice applies a memoryless strategy because, if Bob applies a memoryless strategy, there is always a memoryless strategy for Alice that gets to the winning state as fast as possible. Then, by Proposition 8, we get that both Alice and Bob use S so the play is the sequence given in Definition 4 whose length is 2^n. □

5. An efficient computation of the optimal strategies

The "prefer-one" construction given in Definition 4 characterizes the optimal strategies for both Alice and Bob, as shown in the previous section, but it does not provide an efficient mechanism for computing the next move given the current state. Note that a naive implementation of the construction proposed in Definition 4 requires an order of 2^n memory and an order of 2^n time.

In this section, we provide an efficient way to compute the strategies. Specifically, we identify an efficiently computable function that measures the progress of the game towards Alice's win. Using this function we will describe an efficiently computable procedure for Alice to make her choices such that progress is maximized and an efficiently computable procedure for Bob to hold back progress as much as possible.

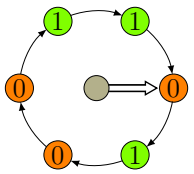
The ranking function we propose is given in the following two definitions:

Definition 10. For a state $s = \langle b_0, \ldots, b_{n-1} \rangle$ and $k \in \{0, \ldots, n-1\}$ let $rank(s, k) = \sum_{i=1}^{n} b_{k-i} 2^i$ where index arithmetic is modulo n.

The index k that maximizes the rank is called the *head* of the state and the maximal rank is called the *rank* of the state:

Definition 11. For a state s let $head(s) = \arg\max_k rank(s,k)$ and $rank(s) = rank(s, head(s))$.

For example, the rank of the state



$= \quad \langle 1,1,0,0,1,0\rangle$

is the number $110010_2 = 50_{10}$, where subscripts denote number bases. In this state, the handle points at the head, i.e., $head(s) = n-1$. It does not have to be that way.

Using this notation, we can describe an easy to compute winning strategy for Alice. The trick in this strategy is that Alice can force that the rank of the state increases repeatedly until the game arrives at the state with the maximal rank, $\langle 1,\ldots,1\rangle$, from which Alice can win in n turns by putting 0 in all the bits. To describe the strategy we need one last bit of notation:

Definition 12. Let $tail(s)$ be the index of the last nonzero bit before $head(s)$ in cyclic order: $tail(s) = \max\{i \le head(s) \colon b_i \neq 0\}$ where index arithmetic is modulo n (i.e., we add n to negative numbers).

For example, for the above state $tail(s) = n-2$, corresponding to the digit at five o'clock in the circular depiction of state. In general, in the circular depiction of states, $tail(s)$ is the first nonzero bit after (inclusive) $head(s)$ going clockwise.

Now, we can state the winning strategy for Alice: put zero if and only if the handle is on $tail(s)$, i.e., if and only if $tail(s) = n-1$. Formally:

Proposition 13. *The only winning memoryless strategy for Alice is given by*

$$A(s) = \begin{cases} 0 & \text{if } tail(s) = n-1; \\ 1 & \text{otherwise.} \end{cases}$$

Proof. If we are in a state were $tail(s) = n-1$, i.e., the handle in the circular depiction of the game points at the $tail(s)$ bit, then Alice's strategy effectively starts from the least significant nonzero bit of $rank(s)$ and puts zero in it (we are talking about the rank of the state after Alice's move). If Bob is not going to switch any of the zeroes in this number to one (and, by that, increase the rank of the state), we are going to arrive at the state 0^n and Alice wins. Therefore, we get that either the rank of the state increases repeatedly until we reach the state 1^n from which Alice wins in n turns or Bob lets the game arrive at the state 0^n before that. Either way, Alice wins. □

Proposition 14. *The only memoryless strategy for Bob that forces 2^n turns before Alice's win is:*

$$B(b_0, \ldots, b_{n-2}, 0) = \begin{cases} 1 & \text{if } tail(b_0, \ldots, b_{n-2}, 1) = n-1; \\ 0 & \text{otherwise.} \end{cases}$$

Proof. With this strategy Bob minimizes the inevitable increase of rank. □

The only nontrivial computation in the strategies given in Proposition 13 and in Proposition 14 is the operator $tail(s)$ which can directly computed in $O(n)$ time and memory.

6. An efficiently computable shift-rule for the reversed prefer-one sequence

Based on the strategies given in in Proposition 13 and in Proposition 14 and the fact that the game play is exactly the reverse of the prefer-one sequence, we can give the following shift-rule for this sequence:

$$shift\_rule(b_0, \ldots, b_{n-2}, b_{n-1}) = \begin{cases} \langle 1 - b_{n-1}, b_0, \ldots, b_{n-2} \rangle & \text{if } tail(b_0, \ldots, b_{n-2}, 1); \\ \langle b_{n-1}, b_0, \ldots, b_{n-2} \rangle & \text{otherwise.} \end{cases}$$

Since $tail(s)$ can be computed with $O(n)$ time and memory complexity, we say that this shift-rule is efficiently computable.

7. Games for all De Bruijn sequences

A generalization of the prefer-one construction, to construct all binary De Bruijn of ordern n, was proposed by L. R. Ford [4]. The, so called, Ford algorithm consists of a set of rules that determine which symbol is tried first after a word of length $n-1$ appears. In this section, we follow this direction towards proposal of a game for each De Bruijn sequence such that the optimal strategies for Bob and for Alice for the game generate the sequence. This is a generalization of what we showed above. We follow Ford in that we apply the same set of rules to decide, given the first $n-1$ bits of the game state, whether one in the last bit means Alice's or Bob's turn. For example, the game given in Definition 1 correspond to the constant rule that says that if the last bit is one its always Alice's turn.

Definition 15. Given a function $f\colon \{0,1\}^{n-1} \to \{0,1\}$, an n-bit generalized shift game is played by two players, Alice and Bob, as follows. The initial state of the game is the n-tuple $s_0 = \langle 0, \ldots, 0\rangle$ and a play of the game consists of the sequence $\langle s_t\rangle_{t=0}^{\infty}$, $s_t \in \{0,1\}^n$, such that if $s_t = \langle b_1, \ldots, b_n\rangle$ then

$$s_{t+1} = \begin{cases} \langle A(s_0, \ldots, s_t), b_1, \ldots, b_{n-1}\rangle & \text{if } f(b_1, \ldots, b_{n-1}) \oplus b_n = 1; \\ \langle B(s_0, \ldots, s_t), b_1, \ldots, b_{n-1}\rangle & \text{if } f(b_1, \ldots, b_{n-1}) \oplus b_n = 0; \end{cases}$$

where $A\colon (\{0,1\}^n)^* \to \{0,1\}$ is Alice's strategy and $B\colon (\{0,1\}^n)^* \to \{0,1\}$ is Bob's strategy. Alice wins the game if there exists $t > 0$ such that $s_t = s_0$ and Bob wins otherwise.

This is a direct generalization of Definition 1. Specifically, if we choose f to be the constant function that assigns 0 to all inputs, we get exactly the same game as in Definition 1. However, by choosing other functions, we can get all binary De Bruijn sequences as solutions of the game in Definition 15, as we show next.

Proposition 16. *Any binary De Bruijn sequence can be viewed as a solution of a generalized shift game for an appropriate choice of a function f.*

Proof. Let $\beta_1, \beta_2, \ldots, \beta_{2^n}$ be a binary De Bruijn sequence of order n. Let $f\colon \{0,1\}^{n-1} \to \{0,1\}$ be such that $f(b_1, \ldots, b_{n-1}) = 0$ if and only if the subsequence $b_1, \ldots, b_{n-1}, 1$ appears before the subsequence $b_1, \ldots, b_{n-1}, 0$ in the sequence $\beta_1, \beta_2, \ldots, \beta_{2^n}$. I is easy to show, using aguments similar to those used for the shift game and the reversed prefer-max sequence, that the

sequence $\langle s_t \rangle_{t=0}^{2^n-1}$ of n-order binary vectors where, for all $t = 0, \ldots, 2^n - 1$, $s_t = \langle \beta_t, \beta_{t+1}, \ldots, \beta_{t+n-1} \rangle$ constitutes a solution to the generalized shift game with the function f. □

8. Conclusions and future work

We have established a connection between combinatorial games and De Bruijn sequences and demonstrated how it can be used to devise an efficiently computable a shift-rule for a De Bruijn sequence.

We also defined games for all other De Bruijn sequences. A future work can be to solve more of these games and to use the solutions to give efficiently computable shift-rules for other De Bruijn sequences. For example, one may start with the prefer-opposite sequence [1] which gives an intuitive game that may be solvable with methods similar to those used here.

Another possible research direction is to extend the discussion beyond binary sequences towards fining efficiently computable shift-rules for non-binary De Bruijn sequences (see, e.g., [6]). For example, one may start with the prefer-max De Bruijn sequence which is a generalization of the prefer-one sequence discussed here.

References

[1] A. M. Alhakim. A simple combinatorial algorithm for de bruijn sequences. *American Mathematical Monthly*, 117(8):728–732, 2010.

[2] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays*, volume 3. AK Peters Natick, 2003.

[3] N. G. de Bruijn. *Acknowledgement of priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2n zeros and ones that show each n-letter word exactly once*. Department of Mathematics, Technological University, 1975.

[4] L. Ford. A cyclic arrangement of m-tuples, report no. *P-1071, Rand Corporation, Santa Monica, California*, 1957.

[5] M. H. Martin. A problem in arrangements. *Bull. Amer. Math. Soc.*, 40:859–864, 1934.

[6] J. Sawada, A. Williams, and D. Wong. A simple shift rule for k-ary de bruijn sequences. *Discrete Mathematics*, 340(3):524 – 531, 2017.

[7] G. Weiss. A combinatorial game approach to state nullification by hybrid feedback. In *Decision and Control, 2007 46th IEEE Conference on*, pages 4643–4647. IEEE, 2007.