De Bruijn Sequences: from Games to Shift-Rules to a Proof of the Fredricksen-Kessler-Maiorana Theorem

Gal Amram   Amir Rubin   Yotam Svoray   Gera Weiss

Submitted: Nov 20, 2024; Accepted: Jun 19, 2025; Published: Jul 18, 2025

Abstract

We present a combinatorial game and propose efficiently computable optimal strategies. We then show how these strategies can be translated to efficiently computable shift-rules for the well known prefer-max and prefer-min De Bruijn sequences, in both forward and backward directions. Using these shift-rules, we provide a new proof of the well known theorem by Fredricksen, Kessler, and Maiorana on De Bruijn sequences and Lyndon words.

Mathematics Subject Classifications: 68R15, 68R10, 91A68

1 Introduction

A De Bruijn sequence of order n over the alphabet $[k] = \{0, \dots, k-1\}$ is a cyclic sequence of length k^n such that every possible word of length n over this alphabet appears exactly once as a subword [1]. In this work we focus on two of the most famous De Bruijn sequences called the prefer-max and the prefer-min sequences [2, 3], obtained by starting with 0^n, for the prefer-max, or $(k-1)^n$, for the prefer-min, and adding to each prefix of maximal/minimal value in $[k]$ such that the suffix of length n does not appear as a subword of the prefix.

A shift-rule of a De Bruijn sequence is a mapping shift: $[k]^n \to [k]^n$ such that, for each word $w = \sigma_1 \dots \sigma_n$, shift$(w)$ is the word $\sigma_2 \dots \sigma_n \tau$ where τ is the symbol that follows w in the sequence (for the last word τ is the first symbol in the sequence). Shift rules of De Bruijn sequences are used in various fields such as cryptography and electrical engineering (for examples, see [4, 5, 6]). Amram et al. [7] proposed an efficiently computable[1] shift-rule for the sequence defined by the concatenation of an ordered list of words called Lyndon words. They then conclude that their shift-rule generates the prefer-min sequence, using the well known FKM theorem by Fredricksen, Kessler, and Maiorana [8, 3] which says that this concatenation is in fact the prefer-min sequence.

Department of Computer Science, Ben-Gurion University of The Negev (galamra@cs.bgu.ac.il, amirrub@cs.bgu.ac.il, ysavorai@post.bgu.ac.il, geraw@cs.bgu.ac.il).

[1] In this paper the term “efficiently computable” always means $O(n)$ time and space

 https://doi.org/10.37236/13603

When $k = 2$, the prefer-max sequence is called prefer-one, for obvious reasons. Weiss [9] proposed a combinatorial game such that a play of two optimal players yields the prefer-one sequence. He also developed efficiently computable optimal strategies for both players in this game and used these to propose an efficiently computable shift rule for the prefer-one sequence.

The first result in this paper is a generalization of Weiss's result to larger alphabets. Specifically, we present a two-player combinatorial game over arbitrary alphabet ($k \geqslant 2$) such that if both players play optimally the play of the game gives the prefer-max sequence. Independently, DiMuro [10] published a text that describes the same game in a slightly different way, named "The Warden Game", and discussed the connection between its game tree and the lexicographically minimal De Bruijn sequence. Note that DiMuro did not present an efficiently computable strategy for the game, and thus his work does not directly give an efficient algorithm for generating the sequence.

Our second result is efficiently computable optimal strategies for both players in the new game. These strategies are a generalization of the strategies proposed by Weiss.

The third result in this paper is the development of efficiently computable optimal strategies which can be used to construct an efficiently computable shift-rules for the prefer-max and prefer-min sequences in both forward and backward directions. Note that reversing the direction is not an issue in the binary case, since one can simply try both options, but it is an issue in the general case since trying all options adds a factor of k to the complexity.

Finally, we show that our shift-rule is equivalent to the one presented by Amram et al. [7] and use this fact to prove the FKM theorem. This is straight-forward since our shift-rule generates the prefer-min sequence directly while Amram's shift-rule generates the sequence produced by the concatenation of the Lyndon words.

An implementation of this work can be found online.[2]

2 Preliminaries

The directed De Bruijn digraph of order n over the alphabet $[k]$ is the digraph whose vertices are the words of length n over the alphabet $[k]$ (i.e. the set $[k]^n$) and whose edges are such that each vertex $v = \tau x$ is connected with directed edges to all vertices in $\{x\sigma \colon \sigma \in [k]\}$.

There is a one-to-one correspondence between De Bruijn sequences and Hamiltonian cycles in the De Bruijn digraph of the same order and alphabet, described in [1] and it is as follows:

1. If, for each i, $w_i = x_i\sigma_i$, and $(w_1, w_2, \dots)$ is an Hamiltonian cycle then $(\sigma_1, \sigma_2, \dots)$ is a De Bruijn sequence.

2. A Hamiltonian cycle can be constructed from a De Bruijn sequence $(\sigma_1, \dots, \sigma_{k^n})$ by visiting the vertex $\sigma_1 \cdots \sigma_n$, then $\sigma_2 \cdots \sigma_{n+1}$ and so on, until we return to where we started.

[2] https://github.com/amirubin87/De-Bruijn-Sequences

In this paper we focus on a specific Hamiltonian cycle in the De Bruijn digraph called the prefer-max cycle (and the corresponding prefer-max De Bruijn sequence).

Definition 1. The (n, k)-prefer-max cycle, $(w_i)_{i=0}^{k^n-1}$, is defined by $w_0 = 0^{n-1}(k-1)$ and if $w_i = \tau x$ then $w_{i+1} = x\sigma$ where σ is the maximal letter such that $x\sigma \notin \{w_0, \ldots, w_i\}$. We denote $w_i \prec w_j$ if $i < j$ in this sequence.

We also consider the (n, k)-prefer-min cycle defined in a similar way, by starting with $w_0 = (k-1)^{n-1}0$ and using the minimal σ instead of the maximal.

Martin [11] proved that the cycle given in Definition 1 is Hamiltonian, i.e., that for $k^n - 1$ steps there is always a σ such that $w\sigma \notin \{w_0, \ldots, w_i\}$. This means that w_{i+1} is well defined. Definition 1 above is demonstrated in Example 2.

Example 2. The $(2, 3)-$prefer-max cycle is given by $002 \to 022 \to 222 \to 221 \to 212 \to 122 \to 220 \to 202 \to 021 \to 211 \to 112 \to 121 \to 210 \to 102 \to 020 \to 201 \to 012 \to 120 \to 200 \to 001 \to 011 \to 111 \to 110 \to 101 \to 010 \to 100 \to 000$.

3 A useful property of the prefer max sequence

Before diving to the specific contribution of this paper, we identify a useful property of the prefer-max cycle.

Lemma 3. *For any words x, y such that $|x|+|y| = n-1$, let $(\sigma_i)_{i=0}^{k-1}$ be such that $(x\sigma_i y)_{i=0}^{k-1}$ is the subsequence of the (n, k)-prefer-max cycle consisting of all the words that begin with x and end with y. Then, there exist some $d \in [k]$ such that the sequence is sorted with the exception of $\sigma_d = 0$. That is,*

$$(\sigma_i)_{i=0}^{k-1} = (k-1, \ldots, k-d, 0, k-d-1, \ldots, 1).$$

Proof. By induction on the length of y. If y is the empty word, the statement is true (with $d = k-1$) from the definition of the prefer-max cycle (Definition 1). For the induction step, assume that for some $t < n-1$ the statement is true for all y of length t. We need to show that $x\sigma_2 y\tau \prec x\sigma_1 y\tau$ for any symbols τ, $0 < \sigma_1 < \sigma_2$ and any word x of length $n-t-2$. Let $v_1 = x\sigma_1 y\tau$, and $v_2 = x\sigma_2 y\tau$. As $\sigma_1 \neq 0$, for each $\psi \in [k]$, $x\sigma_1 y\psi$ is not the first element in the cycle, thus, it has a predecessor. By the definition of the prefer-max cycle $x\sigma_1 y(k-1) \prec x\sigma_1 y(k-2) \prec \cdots \prec x\sigma_1 y0$. Because the predecessor of each of these vertices is in $\{\varphi x\sigma_1 y \colon \varphi \in [k]\}$ we have that $k-\tau$ vertices in this set precede v_1. By the induction hypothesis we get that $\varphi x\sigma_2 y \prec \varphi x\sigma_1 y$, for any φ, and therefore at least $k-\tau$ vertices in $\{\varphi x\sigma_2 y \colon \varphi \in [k]\}$ precede v_1. The follower of each of these vertices is in $\{x\sigma_2 y(k-1), x\sigma_2 y(k-2), \ldots, x\sigma_2 y0\}$ whose members, by definition, appear in decreasing lexicographical order in the prefer-max cycle. Therefore, at least $k-\tau$ vertices in $\{x\sigma_2 y(k-1), x\sigma_2 y(k-2), \ldots, x\sigma_2 y0\}$ precede $x\sigma_1 y\tau$. From this we get that $x\sigma_2 y\tau$ must be before $x\sigma_1 y\tau$. □

Example 4. Consider the sequence from Example 2, and choose $x = 2$ and $y = 0$. For these values, we get the subsequence: $(002, 022, 012)$ that gives us $(0, 2, 1)$ which is sorted if we remove 0 from it.

Observation 5. *For any $x \in [k]^{n-1}$, let $S_1 = (x(k-1), \ldots, x\sigma, \ldots, x0)$ be the subsequence of the (n,k)-prefer-max cycle $(w_i)_{i=0}^{k^n-1}$ consisting of all words that start with x. By Lemma 3, there is a $d \in [k]$ such that $S_2 = ((k-1)x, \ldots (k-d)x, 0x, (k-d-1)x, \ldots, 1x)$ is a subsequence of $\{w_i\}_{i=0}^{k^n-1}$. Hence, the ith element of S_1 is preceded by the ith element of S_2, and every word $w_i = x\sigma, i \neq 0$, is preceded by $w_{i-1} \in \{\sigma x, 0x, (\sigma+1)x\}$.*

Based on this observation, the game that we present in the next section focuses on following the sequence backwards, specifically, on choosing which of the three possible predecessors of a state is chosen.

4 A combinatorial game for the prefer-max sequence

The main object that we analyse in this paper is the following combinatorial game, played between two players, Alice and Bob. A *state* of the game is a word $s \in [k]^n$, and the *initial-state* is 0^n. In each game round, Bob plays first. If the state is $s = w\sigma$, Bob can either set the next state to be $(\sigma+1)w$, or pass control to Alice. Note that if $\sigma = k-1$ Bob cannot increase σ and thus he must pass control to Alice. In case that Bob passes control, Alice gets to choose the next state. She, then, has two options: she can choose the successor state to be either σw, or $0w$.

Bob's goal is to reach an already-seen state $s \neq 0^n$. Alice's goal is to reach 0^n quickly. A play ends in a tie if 0^n is reached only after traversing all k^n-possible states. The next definition formalizes these requirements, and define the notions of strategies, state-progressions and plays.

Definition 6. The (n,k)-shift-game is a two-player combinatorial game defined as follows:

- *Strategies* for the players Alice and Bob, respectively, are functions $A, B\colon [k]^n \to \{0,1\}$ such that $B(x(k-1)) = A(x0) = 0$ for all $x \in [k]^{n-1}$.[3]
- A *state progression* of the game with the strategies A and B is a (finite or infinite) sequence $s_0, s_1, \ldots$ for some $s_0 \in [k]^n$ and if $s_t = x\sigma$ for $x \in [k]^{n-1}$ and $\sigma \in [k]$, then s_t is last in the sequence or:

$$s_{t+1} = \begin{cases} (\sigma+1)x & \text{if } B(s_t) = 1, \\ 0x & \text{if } B(s_t) = 0 \text{ and } A(s_t) = 1, \\ \sigma x & \text{otherwise;} \end{cases}$$

- A (complete) play of the (n,k)-shift-game is a state progression starting with $s_0 = 0^n$ and ending with $s_m \in \{s_0, \ldots, s_{m-1}\}$, such that no prefix of it is a play.
- Alice wins a play $s_0, \ldots, s_m$ if $m < k^n$ and $s_m = 0^n$.

[3]The numbers 1 and 0 that A and B assign to states represent active and passive actions, respectively. Bob's active action is to increase the last symbol by 1, and his passive action is to pass the turn to Alice. Alice's active action is to write 0, wheres her passive action is to copy the last symbol.

- Bob wins a play $s_0, \ldots, s_m$ if $s_m \neq 0^n$.
- A play $s_0, \ldots, s_m$ is a tie if $m = k^n$ and $s_m = 0^n$.

Note that the condition that $A(x0) = 0$ in the first bullet is, in some sense, vacuous since $A(x0) = 1$ produces the same output. We added this requirement to assure uniqueness of strategies. Later, it will be convenient that the choice between active and passive actions is always meaningful.

The fact that not all functions are strategies is not a problem because we can construct new strategies from existing ones using the following fact:

Observation 7. *If S is a strategy for Alice or for Bob and $S' \colon [k]^n \to \{0,1\}$ is a function such that $S'(w) \leqslant S(w)$ for all w, then S' is also a strategy for the same player.*

This game is a generalization of a game defined and analyzed in [9]. There, the game was only defined for a binary alphabet ($k = 2$) and was used to produce an efficient algorithm to construct the prefer-prefer-one sequence. Here, we show how the definition can be extended to a larger alphabet and how this can be used to produce an efficient algorithm for constructing any prefer-max sequence, for any $k \geqslant 2$.

The game that we are proposing was independently proposed by DiMuro in [10]. Compared to Dimuro's paper, the contribution of this paper is an efficiently computable winning strategy for each of the players and a method for using these strategies for efficiently computing a shift rule for De Bruijn sequences.

5 Non-losing strategies for both players

Next, we turn to establishing the connection between the prefer-max cycle and the shift-game. We first define a pair of strategies A^* and B^* such that if both Alice and Bob use the respective strategies, the play of the game follows the prefer-max cycle in reversed order. Then, we show that A^* and B^* are the unique non-losing strategies for Alice and for Bob respectively. This gives us that the efficient implementations of non-losing strategies for both of the players, that we provide in the following sections, can serve as an efficient shift-rule for the cycle.

The strategies A^* and B^* use the prefer-max cycle as an internal 'oracle', as specified in the following definition:

Definition 8. Considering $(w_i)_{i=0}^{k^n-1}$ from Definition 1, let $A^*, B^* \colon [k]^n \to \{0,1\}$ be the strategies for Alice and Bob, respectively, defined by

$$B^*(w_i) = \begin{cases} 1 & \text{if } w_i = x\sigma \wedge w_{i-1} = (\sigma+1)x, \\ 0 & \text{otherwise;} \end{cases}$$

and

$$A^*(w_i) = \begin{cases} 1 & \text{if } w_i = x\sigma \wedge \sigma \neq 0 \wedge (w_{i-1} = 0x \vee B^*(w_i) = 1), \\ 0 & \text{otherwise.} \end{cases}$$

From Definition 1, we can see that A^* and B^* are strategies: (1) By definition, $A^*(x0) = 0$ for all $x \in [k]^{n-1}$; (2) If $w_i = x(k-1)$ then w_{i+1} cannot be kx because this word is not in $[k]^n$, thus $B^*(w_i)$ must be zero.

The essence of both strategies is to follow the prefer-max cycle. B^* maps an element $x\sigma$ on the cycle to 1 if its predecessor is $(\sigma+1)x$. This means that Bob forces the next state to be $(\sigma+1)x$ in these cases and, by the rules of the game Alice will not be able to object. In all other cases, A^* maps $x\sigma$ to 1 when its predecessor is $0x$. If σ is zero, there is no difference to the state of the game if Alice chooses to play or not, and therefore the condition $\sigma \neq 0$ in the definition of A^* serves the technical role of forcing A^* to be a strategy. Additionally, the condition that $A^*(w_i) = 1$ when $B^*(w_i) = 1$ was added to make it a winning strategy against any strategy that Bob may choose, as we show below.

Example 9. For example, for $n = 3$ and $k = 2$, if both players play according to the strategies above, then the resulting play is $000 \to 100 \to 010 \to 101 \to 110 \to 111 \to 011 \to 001 \to 000$ which yields a tie.

We can see that in the example above, the play is exactly the prefer-max cycle in reverse. In general, it is clear from Definition 8 that this is true for all n and k, as stated in the next observation:

Observation 10. *Let $(s_t)_{t=0}^{m}$ be the play of the (n,k)-shift-game when Alice uses the A^* strategy and Bob uses the B^* strategy. Then, $(s_t)_{t=0}^{m-1} = (w_{k^n-t})_{t=1}^{k^n}$ where $(w_i)_{i=0}^{k^n-1}$ is the prefer-max cycle given in Definition 1.*

The next proposition shows that the computation of B^* can be reduced to a computation of A^* over a slightly alternated input. We use this fact to focus only on Alice's strategy, i.e., we'll develop an efficient algorithm to compute A^* and, by the reduction specified in the following proposition, get the same for B^*.

Proposition 11. *$B^*(x\sigma) = A^*(x(\sigma+1))$ for every $x \in [k]^{n-1}$ and $\sigma < k-1$.*

Proof. Let $d\colon [k]^{n-1} \to [k]$ be defined by $d(x) = |\{\tau\colon \tau x \prec 0x\}|$. From Lemma 3 we get that for every $x \in [k]^{n-1}$, and $w_i = x\sigma$:

- if $\sigma < k - d(x) - 1$ then $w_{i-1} = (\sigma+1)x$;
- if $\sigma = k - d(x) - 1$ then $w_{i-1} = 0x$;
- if $\sigma > k - d(x) - 1$ then $w_{i-1} = \sigma x$.

Therefore, by the definition of B^*, $(B^*(x\sigma) = 1) \Leftrightarrow (\sigma < k - d(x) - 1)$, and, by the definition of A^*, for every $\sigma > 0$, $(A^*(x\sigma) = 1) \Leftrightarrow (\sigma \leqslant k - d(x) - 1)$. This means that $B^*(x\sigma) = 1$ if and only if $A^*(x(\sigma+1)) = 1$. □

From Observation 10 we get that if Alice plays according to A^* and Bob plays according to B^*, the game ends with a tie. Our next goal, in propositions 13 and 14, is to show that A^* is the only strategy that wins against any other strategy. Towards this goal, in Lemma 12, we first analyze the options for Bob and Alice in a given state $w_i = x\sigma$. Namely, we consider the state w_{i-1}, that by Observation 10 is the next state in the play of the game when A^* and B^* are used, and compare its position in the sequence relative to the other two options in $\{\sigma x, 0x, (\sigma+1)x\}$.

Lemma 12. *Let $(w_i)_{i=0}^{k^n-1}$ be the (n,k)-prefer-max cycle. If $i > 0$, $w_i = x\sigma$, $x \in [k]^{n-1}$, and $\sigma \in [k]$ then:*

1. $w_{i-1} \in \{\sigma x, 0x, (\sigma+1)x\}$;
2. *if $w_{i-1} = (\sigma+1)x$ then $0x \prec (\sigma+1)x$. If, in addition, $\sigma \neq 0$ then $(\sigma+1)x \prec \sigma x$;*
3. *if $w_{i-1} = \tau x$ for $\tau \in \{0, \sigma\}$, then:*
   - *if $\sigma \neq 0$, then $\tau x \prec \hat{\tau} x$ where* $\hat{\tau} = \begin{cases} \sigma & \text{if } \tau = 0; \\ 0 & \text{if } \tau = \sigma \end{cases}$
   - *if $\sigma < k-1$, then $(\sigma+1)x \prec \tau x$.*

Proof. Item 1 is Observation 5. To prove 2 and 3, we consider the two subsequences of the prefer-max cycle presented in Observation 5: $S_1 = (x(k-1)), \ldots, x\sigma, \ldots, x0)$ and $S_2 = ((k-1)x, \ldots (k-d)x, 0x, (k-d-1)x, \ldots, 1x)$. The ith element of S_1 is preceded by the ith element of S_2. Therefore, (1) if $\sigma > k-d-1$, $x\sigma$ is preceded by σx; (2) if $\sigma = k-d-1$, $x\sigma$ is preceded by $0x$; (3) and otherwise if $\sigma < k-d-1$, then $x\sigma$ is preceded by $(\sigma+1)x$. The first inequality in 2 follows from cases (2) and (3) and the second inequality is a direct result of Lemma 3.

For the first bullet in 3, assume $\sigma \neq 0$. If $\tau = \sigma$, then $\sigma > k-d-1$ thus $\tau x = \sigma x \prec 0x = \hat{\tau} x$. Similarly, if $\tau = 0$, then, as mentioned in case (2) above, $\sigma = k-d-1$. Following subsequence S_2 we have that $\tau x = 0x \prec (k-d-1)x = \sigma x = \hat{\tau} x$.

Lastly, to prove the second bullet in 3, assume $w_{i-1} \neq (\sigma+1)x$ and $\sigma < k-1$. Therefore, either (1) or (2) holds and thus $\sigma \geqslant k-d-1$. If $\sigma > k-d-1$ then by (1) $\tau = \sigma$ and $(\sigma+1)x \prec \sigma x$. Otherwise, if $\sigma = k-d-1$, then by (2) we have that $\tau = 0$. By the definition of d, $(\sigma+1)x = (k-d)x \prec 0x$. □

The next proposition shows that we achieved the purpose of the game: the strategies that generate the prefer-max cycle as a play, A^* and B^*, are optimal strategies for both players:

Proposition 13. *If Alice applies the strategy A^* she wins against any strategy that Bob may apply which is not B^* and gets a tie against B^*.*

Proof. First, from Observation 10 we know that if the players play by the strategies A^* and B^* respectively we get a tie.

Second, let $B \neq B^*$ be a strategy played by Bob and i be such that $B(w_i) \neq B^*(w_i)$ and $w_i = x\sigma$: Let w_t be the state that follows w_i in the game played using A^* and B. We show, using the case analysis below, that $t < i-1$:

- When $B(w_i) = 0$, $B^*(w_i) = 1$ and $A^*(w_i) = 1$. By the definition of B^* we have that $w_{i-1} = (\sigma+1)x$. By the definition of the game we have that $w_t = 0x$. By Lemma 12 (item 2), we have that $w_t = 0x \prec (\sigma+1)x = w_{i-1}$ which gives us that $t < i-1$.

- When $B(w_i) = 0$, $B^*(w_i) = 1$ and $A^*(w_i) = 0$. By the definition of B^* we have that $w_{i-1} = (\sigma+1)x$. By the definition of A^* we have that $\sigma = 0$, and so, by the definition of the game $w_t = 0x$. Again, by Lemma 12 (item 2), we have that $w_t = 0x \prec (\sigma+1)x = w_{i-1}$ which gives us that $t < i-1$.

- When $B(w_i) = 1$, $B^*(w_i) = 0$ and $A^*(w_i) = 1$. By the definition of A^* and B^* we have that $w_{i-1} = 0x$. As B is a strategy, $\sigma < k-1$. By the definition of the game $w_t = (\sigma+1)x$. By Lemma 12 (item 3, second bullet, $\tau = 0$), we have that $w_t = (\sigma+1)x \prec 0x = w_{i-1}$ which gives us that $t < i-1$.

- When $B(w_i) = 1$, $B^*(w_i) = 0$ and $A^*(w_i) = 0$. By the definition of A^* and B^* we have that $w_{i-1} = \sigma x$. As B is a strategy, $\sigma < k-1$. By the definition of the game $w_t = (\sigma+1)x$. By Lemma 12 (item 3, second bullet, $\tau = \sigma$), we have that $w_t = (\sigma+1)x \prec \sigma x = w_{i-1}$ which gives us that $t < i-1$.

Together with the fact that $t = i-1$ if $B(w_i) = B^*(w_i)$, we get that A^* and B produce a play that is a strict subsequnce of the prefer-max cycle. In particular, this play ends in the state $s = 0^n$ and it length is shorter than k^n states, i.e., Alice wins. □

A direct result of the previous proposition is that B^* is the unique non-losing strategy for Bob. We next show the same for A^*:

Proposition 14. *A^* is the only non-losing strategy for Alice.*

Proof. We show that for any strategy $A \neq A^*$ there is some strategy B such that B wins against A.

Let B be defined by

$$B(w_i) = \begin{cases} B^*(w_i) & \text{if } A(w_i) = A^*(w_i), \\ 0 & \text{otherwise.} \end{cases}$$

Because $B(w_i) \leqslant B^*(w_i)$, for any w_i, it is a strategy for Bob by Observation 7. Let i be the maximal integer such that $A(w_i) \neq A^*(w_i)$. By Observation 10, A and B produce a play with the prefix $w_{k^n-1}, \ldots, w_i$. We focus on the step following this prefix and argue that the next state in that play is some w_t where $t \geqslant i$, in which case Bob wins.

Write $w_i = x\sigma$ and note that, since $A(x\sigma) \neq A^*(x\sigma)$ we have that $\sigma \neq 0$. Write $w_{i-1} = \tau x$. We distinguish between two cases.

1. If $\tau \in \{0, \sigma\}$ then $w_t = \hat{\tau}x$ where $\hat{\tau} = |\tau - \sigma|$ and, by Lemma 12, item 3, first bullet, $w_{i-1} = \tau x \prec \hat{\tau}x = w_t$.

2. $\tau = \sigma + 1$. By definition $A^*(w_i) = 1$ and thus $A(w_i) = 0$. Hence, the next state is $w_t = \sigma x$, and Lemma 12, item 2 gives us that $w_{i-1} = (\sigma+1)x \prec \sigma x = w_t$.

Now, to complete the proof, we show that $w^t \neq 0^n$. $w^t = 0^n$ can only occur if $w_i = 0^n\sigma$. First, consider the case in which $\sigma = k-1$. In this case, $w_{i-1} = 0^n$ and thus $A^*((0^{n-1}(k-1)) = 1$. Hence, $A(0^{n-1}(k-1)) = 0$ and thus $w_t = (k-1)0^{n-1} \neq 0^n$.

We turn to deal with the general case for $w_i = 0^{n-1}\sigma$, where $\sigma \neq k-1$ (and also recall that $\sigma \neq 0$). Assume towards contradiction that $w_t = 0^n$.

Since 0^n comes last in the prefer max cycle, by Lemma 3, the following two are subsequences of the prefer-max cycle:

- $(0^{n-1}(k-1), \cdots, 0^{n-1}1, 0^{n-1}0)$
- $((k-1)0^{n-1}, \cdots, 10^{n-1}, 00^{n-1})$

Because the element $0^{n-1}(k-1)$ is the first in the sequence it has no predecessor. The predecessor of any other element in the first subsequence, $w = 0^{n-1}\sigma$ is $(\sigma+1)0^{n-1}$, a member of the second subsequence. Specifically, this is true for w_i, and so $A^*(w_i) = 1$ by the definition of A^*. Thus $A(w_i) = 0$ which contradicts the fact that $w_t = 0^n$.

We showed that state $w_t \in \{w_{k^n-1}, \ldots, w_i\}$. Consequently, strategies A and B produce the play: $(w_{k^n-1}, \ldots, w_i, w_t)$, $w_t \neq 0^n$, in which Bob wins. □

In the next section we develop an efficiently computable rule for a non-loosing strategy, $A^\dagger$, for Alice. Using the above uniqueness property of A^*, we conclude that $A^\dagger = A^*$, i.e., that we can compute A^* efficiently.

6 The $A^\dagger$ strategy for Alice

We turn now to formalizing a strategy for Alice based only on a direct analysis of the current state of the game (without locating the state in the prefer-max sequence). The idea is to analyze the states as numbers in base k. More precisely, we consider the equivalence classes of states under cyclic rotation and rank them according to the highest number in base k in a class. Note that this number can only increase when Bob plays. It decreases when Alice plays and kept constant when both players pass. We show that Alice can play in a way such that Bob is forced to increase the value more than she decreases it, i.e., Alice can force the existence of a monotonically increasing subsequence of states. Therefore, the game eventually reaches the state $(k-1)^n$ (the state with the maximal value), from which Alice can play n consecutive steps and reach her goal - the state 0^n.

Towards this goal, we define the value of states as follows. The function *val* reads the state as a number in base k, and the function val^* assigns to each state the highest value in its equivalence class:

Definition 15. For a state $s = \sigma_0 \cdots \sigma_{n-1} \in [k]^n$ let $val(s) = \Sigma_{i=0}^{n-1} \sigma_{n-1-i} \cdot k^i$ and $val^*(s) = \max val(yx) : s = xy\}$.

Example 16. $val(120) = 0{\cdot}1 + 2{\cdot}3 + 1{\cdot}3^2 = 15$, $val(201) = 1{\cdot}1 + 0{\cdot}3 + 2{\cdot}3^2 = 19$ and $val(021) = 1{\cdot}1 + 2{\cdot}3 + 0{\cdot}3^2 = 7$, so $val^*(120) = \max\{15, 19, 7\} = 19$.

Definition 15 is related to the known notion of *Lyndon words* [12, 13] - non-periodic words that are lexicographically least among their rotations. Specifically, considering the function $neg\colon [k]^+ \to [k]^+$ defined by $neg(\sigma_0 \cdots \sigma_l) = (k-1-\sigma_0)\cdots(k-1-\sigma_l)$, the relation is: the state $neg(s)$ is a Lyndon word if and only if it is non-periodic and $val^*(s) = val(s)$.

Based on Definition 15, we propose the following strategy for Alice:

Definition 17. Let $A^\dagger\colon [k]^n \to \{0,1\}$ be the strategy for Alice defined by

$$A^\dagger(0^l w\sigma) = \begin{cases} 1 & \text{if } \sigma \neq 0 \text{ and } val^*(0^l w\sigma) = val(w\sigma 0^l); \\ 0 & \text{otherwise.} \end{cases}$$

for any word w that doesn't start with 0.

It is easy to see that $A^\dagger$ is a strategy for Alice, i.e., $A^\dagger(x0) = 0$ for every $x \in [k]^{n-1}$.

Example 18. $A^\dagger(120) = A^\dagger(012) = 0$, $A^\dagger(201) = 1$.

Note that in the first and the third cases the number of leading zeros is $l = 0$, whereas in the second $l = 0$.

We make use of the following monotony property of $A^\dagger$:

Proposition 19. *For any $x \in [k]^{n-1}$ and $\sigma \in [k] \setminus \{0\}$, $A^\dagger(x\sigma) \geqslant A^\dagger(x\tau)$ for all $\tau > \sigma$.*

Proof. Let $x = 0^l w$ where the first symbol in w is not 0. We need to show that if $val(w\tau 0^l) = val^*(w\tau 0^l)$, then $val(w\sigma 0^l) = val^*(w\sigma 0^l)$. To this end, we show that the value of any rotation of $x\sigma 0^l$ is not greater than $val(x\sigma 0^l)$. The claim clearly holds for rotations that start with 0, thus we consider a partition $x = y_1 y_2$, and a rotation $y_2\sigma 0^l y_1$. By assumption, $val(y_2\tau 0^l y_1) \leqslant val(y_1 y_2 \tau 0^l)$, i.e., $\Delta_\tau = val(y_1 y_2 \tau 0^l) - val(y_2 \tau 0^l y_1) \geqslant 0$. We turn to analyze $\Delta_\sigma = val(y_1 y_2 \sigma 0^l) - val(y_2 \sigma 0^l y_1)$, as required, we get, by the definition of val, that $\Delta_\sigma = \Delta_\tau + (\tau - \sigma)(k^{l+|y_1|} - k^l) > 0$ which means that $val(y_2\sigma 0^l y_1) < val(y_1 y_2 \sigma 0^l)$ as required. □

A fact that plays a key role in Section 8 is that Definition 17 is related to the predicate *head* given in Definition 2 in [7]) as follows:

$$head((k-1)^l w\sigma) = \begin{cases} true & \text{if } \sigma \neq k-1 \text{ and } w\sigma(k-1)^l \text{ and is} \\ & \text{lexicographically minimal among its rotations;} \\ false & \text{otherwise.} \end{cases}$$

Specifically, the relation is:

Proposition 20. *$head(neg(s)) \Leftrightarrow A^\dagger(s) = 1$.*

Proof. $A^\dagger(s) = 1 \Leftrightarrow s = 0^l w\sigma$ where $\sigma \neq 0$ and $w\sigma 0^l$ is lexicographically largest among s's rotations $\Leftrightarrow neg(s) = (k-1)^l neg(w) neg(\sigma)$ where $neg(\sigma) \neq k-1$, and $neg(w)neg(\sigma)(k-1)^l$ is lexicographically smallest among $neg(s)$'s rotations $\Leftrightarrow head(neg(s))$. □

7 $A^\dagger$ is a non-losing efficiently computable strategy for Alice

In this section we propose an algorithm for efficient computation of the $A^\dagger$ strategy for Alice, and show that it is a non-losing strategy. Using the uniqueness of A^*, this leads us to the conclusion that $A^\dagger = A^*$, therefore we can efficiently compute A^*. Moreover, by Proposition 11, the same applies to B^*.

Proposition 21. *$A^\dagger(s)$ can be computed in $O(|s|)$ time and memory.*

Proof. Let $s = \sigma w$ where $\sigma \in [k]$ and $w \in [k]^n$. In order to compute $A^\dagger(s)$, we need to find $s^* \in \arg\max\{val(s')\colon s' \text{ is a rotation of } s\}$. We can compute the value of $val(\sigma x)$ based on $val(x\sigma)$ in $O(1)$ time and memory using the equation $val(\sigma x) = (val(x\sigma) - \sigma)/k + \sigma \cdot k^n$. Therefore, we can extract s^* in $O(n)$ time and memory. Now, given s^*, we can compute $A^\dagger(s)$ in $O(n)$ time and space using: $A^\dagger(s) = 1 \iff s^* = 0^l w\sigma$ and $s = w\sigma 0^l$ where $\sigma \neq 0$. □

The proof of the following proposition is a mathematical formulation of the intuition stated before the definition of $A^\dagger$. Specifically, it explains in detail how Alice forces Bob to increase val^* more than she decreases it and how this drives the game, if Bob plays optimally, to the state $(k-1)^n$ from which Alice wins in n steps.

Proposition 22. *$A^\dagger$ is a non-losing strategy for Alice.*

Proof. We consider an infinite state progression $s_0, s_1, \dots$ when Alice plays according to $A^\dagger$, Bob plays a strategy B, and $s_0 = 0^n$. We show that Bob does not win, regardless of the chosen strategy B. Bob wins against $A^\dagger$ iff, excluding s_0, 0^n does not appear in the sequence. To show that $0^n = s_i$ for some $i > 0$, we assume otherwise and prove the following:

> There exists a subsequence of the state-progression, $s_{i_0}, s_{i_1}, \dots$ such that $val^*(s_{i_0}) < val^*(s_{i_1}) < \cdots$.

Clearly, this leads to a contradiction as val^* cannot infinitely grow.

We start with a few technical claims.

Claim 23. *If $val(x\tau 0^l) = val^*(x\tau 0^l)$, then $val(x00^l) = val^*(x00^l)$.*

Proof of Claim 23. Repeat the argument in the proof of Proposition 19 for the case $\sigma = 0$. □

Claim 24. *Let $t_1 < t_2$ be such that $A^\dagger(s_{t_1}) = 1$, $t_2 - t_1 \leqslant n$, and $B(s_t) = 0$ for every t in the range $t_1 \leqslant t < t_2$. Write $s_{t_1} = xy$ where $|y| = t_2 - t_1$. Then, $s_{t_2} = 0^{t_2 - t_1} x$.*

Proof of Claim 24. Let $w(\tau+1)0^l$ be the rotation of s_{t_1} that satisfies $val^*(s_{t_1}) = val(w(\tau+1)0^l)$. Therefore, as $A^\dagger(s_{t_1}) = 1$, $s_{t_1} = 0^l w(\tau+1)$. Write $s_{t_1} = xy$ such that $|y| = t_2 - t_1$. First, we prove only for the case $|y| \leqslant n - l$. Hence, we can write $s_{t_1} = xy = 0^l zy$. Recall that $val^*(s_{t_1} = 0^l zy) = val(w(\tau+1)0^l = zy0^l)$ and hence, $val^*(zy0^l) = val(zy0^l)$.

Write $y = \sigma_m \sigma_{m-1} \cdots \sigma_1$. We prove by induction that for $i \in \{0, 1, \ldots, m\}$, $s_{t_1+i} = 0^{l+i} z\sigma_m \cdots \sigma_{i+1}$.

The induction hypothesis holds vacuously for $i = 0$. For the induction step, assume that $s_{t_1+i} = 0^{l+i} z\sigma_m \cdots \sigma_{i+1}$. If $\sigma_{i+1} = 0$, the claim holds by the definition of a strategy for Alice. Otherwise, as $val(z\sigma_m \cdots \sigma_1 0^l) = val^*(z\sigma_m \cdots \sigma_1 0^l)$, by Claim 23, also $val(z\sigma_m \cdots \sigma_{i+1} 0^{l+i}) = val^*(z\sigma_m \cdots \sigma_{i+1} 0^{l+i})$. Consequently, $A^\dagger(0^{l+i} z\sigma_m \cdots \sigma_{i+1}) = 1$ and thus $s_{t_1+i+1} = 0^{l+i+1} z\sigma_m \cdots \sigma_{i+2}$. As a result, $s_{t_2} = s_{t_1+|y|} = 0^{l+|y|} z = 0^{|y|} x$, as required.

Now, the case $|y| > l$ is argued as follows: As proved so far, $s_{t_1+(n-l)} = 0^n$. Hence, as long as Bob outputs 0, the state remains 0^n. □

Claim 25. *Let t_1 be such that for every t in the range $t_1 \leqslant t < t_1 + 2n$, $B(s_t) = 0$. Then, for some t in that range, $s_t = 0^n$.*

Proof of Claim 25. If $s_{t_1} = 0^n$ we are done, and otherwise, let $w(\sigma+1)0^l$ be the rotation of s_{t_1} that satisfies $val^*(s_{t_1}) = val(w(\sigma+1)0^l)$. Hence, $0^l w(\sigma+1)$ is the only rotation of s_{t_1} on which $A^\dagger$ outputs 1. As Bob keeps passing the turn to Alice, after i-steps for some $i < n$, the state progression reaches a state $s_{t_1+i} = 0^l w(\sigma+1)$. Hence, by Claim 24, $s_{t_1+i+n} = 0^n$. □

Claim 26. *If $t_1 < t_2$ are such that $B(s_{t_1}) = B(s_{t_2}) = 1$ and $s_t \neq 0^n$ for all $t_1 \leqslant t \leqslant t_2$ then $val^*(s_{t_1+1}) < val^*(s_{t_2+1})$.*

Proof of Claim 26. Note that it is sufficient to prove the claim for the case in which $B(s_t) = 0$ for every t in range $t_1 < t < t_2$.

First, assume that for every t in range $t_1 < t < t_2$, $A^\dagger(s_t) = 0$. In this case, s_{t_2} is a rotation of s_{t_1+1}. As Bob increases one of the symbols of s_{t_2}, the claim follows from the definition of val^*,

Now, assume that for some intermediate state s_t where $t_1 < t < t_2$, $A^\dagger(s_t) = 1$, and take such minimal t. Therefore, s_t is a rotation of s_{t_1+1} and thus

$$val^*(s_{t_1+1}) = val^*(s_t). \tag{1}$$

Let $l \in [n]$, $w \in [k]^{n-l-1}$, and $\sigma \in [k] \setminus \{0\}$ be such that $w\sigma 0^l$ is the rotation of s_{t_1+1} that satisfies $val^*(s_{t_1+1}) = val(w\sigma 0^l)$. Hence, as $A^\dagger(s_t) = 1$, according to the definition of $A^\dagger$, $s_t = 0^l w\sigma$. By Claim 24, there are x, τ, y such that $w = x\tau y$ and $s_{t_2} = 0^{l+1+|y|} x\tau$. Therefore, $s_{t_2+1} = (\tau+1)0^{l+1+|y|} x$.

Now, $val^*(s_t) = val(w\sigma 0^l) = val(x\tau y\sigma 0^l)$. As a result,

$$val^*(s_{t_2+1}) \geqslant val(x(\tau+1)0^{l+1+|y|}) > val(x\tau y\sigma 0^l) = val^*(s_t). \tag{2}$$

The first inequality is by the definition of val^* and the fact that $x(\tau+1)0^{l+1+|y|}$ is a rotation of s_{t_2+1}. The second inequality can be understood if one thinks of val as reading the word in base $[k]$: we decreased a significant digit and increased less significant ones.

Equations 1 and 2 imply that $val^*(s_{t_1+1}) < val^*(s_{t_2+1})$. □

We can finally prove the proposition. Assume towards contradiction that Bob wins with a strategy B. Hence, the obtained play $(s_0, \ldots, s_m)$ does not include 0^n, excluding s_0, and $s_m = s_r$ for some $0 < r < m$. Consider the infinite state progression from $s_0 = 0^n$, $(s_i)_{i=0}^{\infty}$. As Bob wins, excluding s_0, this sequence does not include 0^n. Hence, by Claim 25, there are infinitely many indices i for which $B(s_i) = 1$. Therefore, by Claim 26, there exists an infinite subsequence $(s_{i_j})_{j=1}^{\infty}$ such that $val^*(s_{i_1}) < val^*(s_{i_2}) < \cdots$, in contradiction to the fact that $val^*(w)$ is bounded by k^n. $\square$

Proposition 27. *$A^\dagger = A^*$.*

Proof. From Proposition 22, we have that $A^\dagger$ is a non-losing strategy, and from Proposition 14 we know that A^* is the only non-losing strategy for Alice. $\square$

8 Efficiently computable shift rules

In this section we apply the efficient strategies developed above to propose an efficiently computable shift rules for both prefer min and prefer max De Bruijn sequences, in both directions (backwards and forwards).

We begin with an efficiently computable shift rule for the reverse of the prefer-max cycle:

Theorem 28. *The function*

$$\operatorname{shift}(x\sigma) = \begin{cases} (\sigma+1)x & \text{if } \sigma < k-1 \text{ and } A^\dagger(x(\sigma+1)) = 1; \\ 0x & \text{else, if } A^\dagger(x\sigma) = 1; \\ \sigma x & \text{otherwise.} \end{cases}$$

maps each vertex on the prefer-max cycle to its predecessor and can be computed in $O(n)$ time and memory.

Proof. From Proposition 27, we know that $A^\dagger = A^*$ and by Proposition 11, $A^\dagger(x(\sigma+1)) = B^*(x\sigma)$. By the definition of the game (Definition 6) we have that $s_{t+1} = \operatorname{shift}(s_t)$ where $(s_t)_{t=0}^{k^n-1}$ is the play of the (n,k)-shift-game when Alice uses the A^* strategy and Bob uses the B^* strategy. From Observation 10 we get that $(s_t)_{t=0}^{k^n-1} = (w_{k^n-1-t})_{t=0}^{k^n-1}$, where $(w_t)_{t=0}^{k^n-1}$ is the prefer-max cycle given in Definition 1. $\square$

We next state an efficiently computable shift rule for the prefer-max cycle in the forward direction:

Theorem 29. *The function*

$$\operatorname{shift}^{-1}(\sigma x) = \begin{cases} x(\sigma-1) & \text{if } \sigma > 0 \text{ and } A^\dagger(x\sigma) = 1; \\ x(\max S) & \text{if } \sigma = 0 \text{ and } S = \{\tau \neq 0 \colon A^\dagger(x\tau) = 1\} \neq \emptyset; \\ x\sigma & \text{otherwise.} \end{cases}$$

is the inverse of shift. *It maps each vertex on the prefer-max cycle to its successor.*

Proof. Since shift is a bijection, it is sufficient to show that $\text{shift}^{-1}(\text{shift}(x\sigma)) = x\sigma$. Let $\sigma' x = \text{shift}(x\sigma)$. We split the proof into three cases, in correspondence with the three cases of shift:

1. If $\sigma < k-1$ and $A^\dagger(x(\sigma+1)) = 1$: Since $\text{shift}(x\sigma) = (\sigma+1)x$ we can write $\sigma' = \sigma+1$. Thus $\sigma' > 0$ and $A^\dagger(x\sigma') = 1$, we fall in the first case when computing $\text{shift}^{-1}(\sigma' x)$, and get that $\text{shift}^{-1}(\sigma' x) = x(\sigma' - 1) = x\sigma$.

2. If

   (a) $\sigma = k - 1$ or $A^\dagger(x(\sigma + 1)) = 0$; and

   (b) $A^\dagger(x\sigma) = 1$,

   by the definition of shift, $\sigma' = 0$. Thus, the first case clause of $\text{shift}^{-1}(\sigma' x)$ is false. We next show that we satisfy the conditions for the second case of shift^{-1}, and that $\max S = \sigma$. By (b), we have that $\sigma \in S$, thus S is not empty and we are in the second case of shift^{-1}. If $\sigma = k - 1$, it is the maximal value in S. Otherwise, by (a), $\sigma + 1 \notin S$ thus, by Proposition 19, σ is the maximum of S. Therefore, $\text{shift}^{-1}(\sigma' x) = x\sigma$.

3. If

   (a) $\sigma = k - 1$ or $A^\dagger(x(\sigma + 1)) = 0$; and

   (b) $A^\dagger(x\sigma) = 0$,

   by the definition of shift, $\sigma' = \sigma$. We claim that the first two cases of $\text{shift}^{-1}(\sigma x)$ are both false. First, by (b), the first case of $\text{shift}^{-1}(\sigma x)$ is false. As for the second case, we split the proof to two: If $\sigma > 0$, obviously the second case of $\text{shift}^{-1}(\sigma x)$ is false. When $\sigma = 0$, we are to show that $S = \emptyset$. By the definition of $A^\dagger$, $0 \notin S$. By (a), and as we assumed $\sigma = 0$, we have that $A^\dagger(x(\sigma + 1)) = 0$, thus $1 \notin S$ and by Proposition 19, $S = \emptyset$. Therefore, we are in the third case of shift^{-1}, and thus $\text{shift}^{-1}(\sigma x) = x\sigma$. □

The above two definitions describe shift rules for the prefer-max sequence. For the prefer-min sequence, we can apply *neg* on the vertices of the (n, k)-prefer-max cycle to get the (n, k)-prefer-min cycle (and vice verse). Using this, the next two corollaries gives an efficiently computable shift rule for the prefer-min cycle, both in the backward and forward directions:

Corollary 30. $neg(\text{shift}(neg(s)))$ *maps each vertex on the prefer-min cycle to its predecessor. It can be computed in* $O(n)$ *time and memory.*

Corollary 31. $neg(\text{shift}^{-1}(neg(s)))$ *maps each vertex on the prefer-min cycle to its successor.*

The complexity analysis of the forward direction depends on our ability to efficiently compute $neg(\text{shift}^{-1}(neg(s)))$. As we show in the following proposition, this function was already analyzed by Amram et. al. [7] under the name *next*.

Proposition 32. *The function*

$$next(\sigma x) = \begin{cases} x(\sigma+1) & \text{if } \sigma \neq k-1 \text{ and } head(x\sigma); \\ x(\min S) & \text{else, if } \sigma = k-1 \text{ and } S = \{\tau \neq k-1 \colon head(x\tau)\} \neq \emptyset; \\ x\sigma & \text{otherwise,} \end{cases}$$

defined by Amram et al. [7, Definition 3], satisfies

$$next(\sigma x) = neg(\text{shift}^{-1}(neg(\sigma x))).$$

Proof. Let $\hat{\sigma} = neg(\sigma)$, $\hat{x} = neg(x)$ and $\hat{S} = \{\hat{\tau} \neq k-1 \colon head(\hat{x}\hat{\tau})\}$. Note that, by Proposition 20, $\hat{S} = \{\hat{\tau} : \tau \in S\}$. We need to show the following:

- $\sigma > 0 \wedge A^{\dagger}(x\sigma) = 1$ if and only if $\hat{\sigma} \neq k{-}1 \wedge head(\hat{x}\hat{\sigma})$.
- $\sigma = 0$ and $S \neq \emptyset$ if and only if $\hat{\sigma} = k{-}1$ and $\hat{S} \neq \emptyset$.
- $x(\sigma - 1) = neg(\hat{x}(\hat{\sigma}+1))$.
- $x(\max S) = neg(\hat{x}(\min \hat{S}))$.
- $x\sigma = neg(\hat{x}\hat{\sigma})$.

All of these are true by Proposition 20 and by the definition of *neg*. □

Since Amram et al. [7, Theorem 12] proved that the complexity of *next* is $O(n)$ in both time and memory, we can conclude that:

Corollary 33. *The shift rules given in Theorem 29 and Corollary 31 for the forward direction of both the prefer-max and the prefer-min can be computed in $O(n)$ time and memory.*

9 A proof of the theorem of Fredricksen, Kessler, and Maiorana

Beyond providing efficiently computable rules, our results also cater for a very simple proof of a well known theorem, as follows.

Let $L_0, L_1, \dots, L_m$ be a lexicographic enumeration of all Lyndon words whose length divides n. The main result of [14] (rephrased to simplify the presentation) is:

Theorem 34 (FKM)**.** *$L_0 L_1 \cdots L_m$ is the (n,k)-prefer-min sequence.*

This theorem was first proposed by Fredricksen, Kessler, and Maiorana [14] with a partial proof (only that $L_0 L_1 \cdots L_m$ is a De Bruijn sequence, not that it is the prefer-min sequence). After more than 25 years, Eduardo Moreno provided an alternative proof [15] and then, after an additional ten years, provided more details to this proof with D. Perrin and added a proof of the other part of the statement (that this is the prefer-min sequence) [16]. Here we show another proof of this theorem using the combinatorial game studied in this paper and the following known result:

Theorem 35 (Amram et al. [7])**.** *next is a shift rule for* $L_0 L_1 \cdots L_m$.

Theorem 35 is stated and proved by Amram et al. [7] inside the proof of Theorem 4. Finally, the proof of the FKM theorem follows immediately:

Proof of Theorem 34. By Proposition 32 and Theorem 35, *next* is a shift rule for both the prefer-min cycle and for the concatenation of the Lyndon words. □

Acknowledgements

This research was partially supported by ISF grant #2714/19. We thank the editor and anonymous referees for their constructive input and helpful remarks on the manuscript.

References

[1] N. G. de Bruijn, A combinatorial problem, Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen. Series A 49 (7) (1946) 758.

[2] H. Martin, A problem in arrangements, Bulletin of the American Mathematical Society 40 (12) (1934) 859-864.

[3] H. Fredricksen, I. Kessler, Lexicographic compositions and debruijn sequences, Journal of Combinatorial Theory, Series A 22 (1) (1977) 17-30.

[4] A. Alhakim, M. Akinwande, A recursive construction of nonbinary de Bruijn sequences, Des. Codes Cryptogr. 60 (2) (2011) 155-169. doi:10.1007/s10623-010-9423-7.

[5] S. Annexstein, Generating De Bruijn sequences: An efficient implementation, IEEE Trans. Computers 46 (2) (1997) 198-200. doi:10.1109/12.565596.

[6] M. Miroschnyk, T. Korytchinko, O. Demihev, V. Krylova, D. Karaman, I. Filippenko, Practical methods for de Bruijn sequences generation using non-linear feedback shift registers, 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), IEEE, (2018) 1157-1161.

[7] G. Amram, Y. Ashlagi, A. Rubin, Y. Svoray, M. Schwartz, G. Weiss, An efficient shift rule for the prefer-max de bruijn sequence, Discrete Mathematics 342.1 (2019) 226-232.

[8] H. Fredricksen, Generation of the ford sequence of length 2n, n large, Journal of Combinatorial Theory, Series A 12 (1) (1972) 153-154.

[9] G. Weiss, A combinatorial game approach to state nullification by hybrid feedback, 46th IEEE Conference on Decision and Control, IEEE (2007) 4643-4647.

[10] J. DiMuro, The Warden's de Bruijn sequence, arXiv preprint arXiv:1807.09292 (2018).

[11] M. H. Martin, A problem in arrangements, Bull. Amer. Math. Soc. 40 (1934) 859-864.

[12] R. C. Lyndon, On burnside's problem, Transactions of the American Mathematical Society 77 (2) (1954) 202-215.

[13] J. Berstel, D. Perrin, The origins of combinatorics on words, European Journal of Combinatorics 28 (3) (2007) 996-1022.

[14] M. Fredricksen, J. Maiorana, Necklaces of beads in k colors and k-ary de Bruijn sequences, Discrete Mathematics 23.3 (1978) 207-210.

[15] E. Moreno, On the theorem of Fredricksen and Maiorana about de bruijn sequences, Advances in Applied Mathematics 33 (2) (2004) 413-415.

[16] E. Moreno, D. Perrin, Corrigendum to "On the theorem of Fredricksen and Maiorana about de bruijn sequences" [adv. in appl. math. 33 (2)(2004) 413-415], Advances in Applied Mathematics 62 (2015) 184-187.