Efficient constructions of the Prefer-same and Prefer-opposite de Bruijn sequences

Evan Sala
School of Computer Science, University of Guelph, Canada

Joe Sawada
School of Computer Science, University of Guelph, Canada

Abbas Alhakim
Department of Mathematics, American University of Beirut, Lebanon

Abstract

The greedy Prefer-same de Bruijn sequence construction was first presented by Eldert et al. [*AIEE Transactions* 77 (1958)]. As a greedy algorithm, it has one major downside: it requires an exponential amount of space to store the length 2^n de Bruijn sequence. Though de Bruijn sequences have been heavily studied over the last 60 years, finding an efficient construction for the Prefer-same de Bruijn sequence has remained a tantalizing open problem. In this paper, we unveil the underlying structure of the Prefer-same de Bruijn sequence and solve the open problem by presenting an efficient algorithm to construct it using $O(n)$ time per bit and only $O(n)$ space. Following a similar approach, we also present an efficient algorithm to construct the Prefer-opposite de Bruijn sequence.

1 Introduction

Greedy algorithms often provide some of the nicest algorithms to exhaustively generate combinatorial objects, especially in terms of the simplicity of their descriptions. An excellent discussion of such algorithms is given by Williams [32] with examples given for a wide range of combinatorial objects including permutations, set partitions, binary trees, and de Bruijn sequences. A downside to greedy constructions is that they generally require exponential space to keep track of which objects have already been visited. Fortunately, most greedy constructions can also be constructed efficiently by either an iterative successor-rule approach, or by applying a recursive technique. Such efficient constructions often provide extra underlying insight into both the combinatorial objects and the actual listing of the object being generated.

A *de Bruijn sequence* of order n is a sequence of bits that when considered cyclicly contains every length n binary string as a substring exactly once; each such sequence has length 2^n. They have been studied as far back as 1894 with the work by Flye Sainte-Marie [13], receiving more significant attention starting in 1946 with the work of de Bruijn [7]. Since then, many different de Bruijn sequence constructions have been presented in the literature (see surveys in [15] and [20]). Generally, they fall into one of the following categories: (i) greedy approaches (ii) iterative successor-rule based approaches which includes linear (and non-linear) feedback shift registers (iii) string concatenation approaches (iv) recursive approaches. Underlying all of these algorithms is the fact that every de Bruijn sequence is in 1-1 correspondence with an Euler cycle in a related de Bruijn graph.

Perhaps the most well-known de Bruijn sequence is the one that is the lexicographically largest. It has the following greedy Prefer-1 construction [27].

Prefer-1 construction

1. Seed with 0^{n-1}
2. **Repeat** until no new bit is added: Append 1 if it does not create a duplicate length n substring; otherwise append 0 if it does not create a duplicate length n substring
3. Remove the seed

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

For example, applying this construction for $n = 4$ we obtain the string: ~~000~~ 1111011001010000. Like all greedy de Bruijn sequence constructions, this algorithm has a major downside: it requires an exponential amount of space to remember which substrings have already been visited. Fortunately, the resulting sequence can also be constructed efficiently by applying an $O(n)$ time per bit successor-rule which requires $O(n)$ space [14]. By applying a necklace concatenation approach, it can even be generated in amortized $O(1)$ time per bit and $O(n)$ space [17].

Two other interesting greedy constructions take into account the last bit generated. They are known as the Prefer-same and Prefer-opposite constructions and their resulting sequences are, respectively, the lexicographically largest and smallest with respect to a run-length encoding[1] [3]. The Prefer-same construction was first presented by Eldert et al. [10] in 1958 and was revisited with a proof of correctness by Fredricksen [15] in 1982. Recently, the description of the algorithm was simplified [3] as follows:

Prefer-same construction

1. Seed with length $n-1$ string $\cdots 01010$
2. Append 1
3. **Repeat** until no new bit is added: Append the **same** bit as the last if it does not create a duplicate length n substring; otherwise append the opposite bit as the last if it does not create a duplicate length n substring
4. Remove the seed

For $n = 4$, the sequence generated by this Prefer-same construction is ~~010~~ 1111000011010010. It has run-length encoding 44211211 which is the lexicographically largest amongst all de Bruijn sequences for $n = 4$.

The Prefer-opposite construction is not greedy in the strictest sense since there is a special case when the current suffix is 1^{n-1}. Details about this special case are provided in the next section. The construction presented below produces a shift of the sequence produced by the original presentation in [1]. Here, the initial seed of 0^{n-1} is rotated to the end so the resulting sequence is the lexicographically smallest with respect to a run-length encoding.

Prefer-opposite construction

1. Seed with 0^{n-1}
2. Append 0
3. **Repeat** until no new bit is added:
   - **If** current suffix is 1^{n-1} **then**: append 1 if it is the first time 1^{n-1} has been seen; otherwise append 0
   - **Otherwise**: append the **opposite** bit as the last if it does not create a duplicate length n substring; otherwise append the same bit as the last
4. Remove the seed

For $n = 4$, the sequence generated by this Prefer-opposite construction is ~~000~~ 0101001101111000. The run-length encoding of this sequence is given by 111122143.

To simplify our discussion, let:

- $\mathcal{S}_n$ = the de Bruijn sequence of order n generated by the Prefer-same construction, and
- $\mathcal{O}_n$ = the de Bruijn sequence of order n generated by the Prefer-opposite construction.

[1] The run-length encoding of a string is discussed formally in Section 3.

Unlike the Prefer-1 sequence, and despite the vast research on de Bruijn sequences, $\mathcal{S}_n$ and $\mathcal{O}_n$ have no known efficient construction. For $\mathcal{S}_n$, finding an efficient construction has remained an elusive open problem for over 60 years. The closest attempt came in 1977 when Fredricksen and Kessler devised a construction based on lexicographic compositions [16] that we discuss further in Section 8.

The main results of this paper are to solve these open problems by providing successor-rule based constructions for $\mathcal{S}_n$ and $\mathcal{O}_n$. They generate the respective sequences in $O(n)$ time per bit using only $O(n)$ space. The discovery of these efficient constructions hinged on the following idea:

> Most *interesting* de Bruijn sequence are the result of joining together smaller cycles induced by *simple* feedback shift registers.

The initial challenge was to find such a simple underlying feedback function. After careful study, the following function was revealed:

$$f(w_1w_2\cdots w_n) = w_1 \oplus w_2 \oplus w_n,$$

where $\oplus$ denotes addition modulo 2. We demonstrate this feedback function has nice run-length properties when used to partition the set of all binary strings of length n in Section 4.3. The next challenge was to find appropriate representatives for each cycle induced by f in order to apply the framework from [20] to join the cycles together.

Outline of paper. Before introducing our main results, we first provide an insight into greedy constructions for de Bruijn sequences that we feel has not been properly emphasized in the recent literature. In particular, we demonstrate how all such constructions, which are generalized by the notion of preference or look-up tables [2, 33], are in fact just special cases of a standard Euler cycle algorithm on the de Bruijn graph. This discussion is found in Section 2 which also outlines a second Euler cycle algorithm underlying the cycle joining approach applied in our main result. In Section 3, we present background on run-length encodings. In Section 4, we discuss feedback functions and de Bruijn successors and introduce the function $f(w_1w_2\cdots w_n) = w_1 \oplus w_2 \oplus w_n$ critical to our main results. In Section 5, we present two generic de Bruijn successors based on the framework from [20]. In Section 6 we present our first main result: an efficient successor-rule to generate $\mathcal{S}_n$. In Section 7 we present our second main result: an efficient successor-rule to generate $\mathcal{O}_n$. In Section 8 we discuss the lexicographic composition algorithm from [16] and a related open problem. In Section 9 we discuss implementation details and analyze the efficiency of our algorithms. In Section 10 and Section 11 we detail the technical aspects required to prove our main results. We conclude by presenting directions for future research in Section 12. Implementation of our algorithms, written in C, presented in this paper can be found in the appendices and are available for download at `http://debruijnsequence.org`.

Applications. One of the first instances of de Bruijn sequences is found in works of Sanskrit prosody by the ancient mathematician Pingala dating back to the 2nd century BCE. Since then, de Bruijn sequences and their related theory have a rich history of application. One of their more prominent applications, due to their random-like properties [22], is in the generation of pseudo-random bit sequences which are used in stream ciphers [26]. In particular, linear feedback shift register constructions (that omit the string of all 0s) allow for efficient hardware embeddings which have been classically applied to represent different maps in video games including Pitfall [4]. Another application uses de Bruijn sequences to crack cipher locks in an efficient manner [15]. More recently, the related de Bruijn graph has been applied to genome assembly [6, 28]. Given the vast literature on de Bruijn sequences and their various methods of construction, the more interesting new results may relate to sequences with specific properties. This makes the de Bruijn sequences $\mathcal{S}_n$ and $\mathcal{O}_n$ of special interest since they are, respectively, the lexicographically largest and smallest sequences with respect

to a run-length encoding [3]. Moreover, recently it was noted they have a relatively small discrepancy, which is the maximum absolute difference between the number of 0s and 1s in any substring, when compared to the sequences generated by the Prefer-1 construction [19].

2 Euler cycle algorithms and the de Bruijn graph

The *de Bruijn graph* of order n is the directed graph $G(n) = (V, E)$ where V is the set of all binary strings of length n and there is a directed edge from $u = u_1u_2 \cdots u_n$ to $v = v_1v_2 \cdots v_n$ if $u_2 \cdots u_n = v_1 \cdots v_{n-1}$. Each edge e is labeled by v_n. Outputting the edge labels in a Hamilton cycle of $G(n)$ produces a de Bruijn sequence. Figure 1(a) illustrates a Hamilton cycle in the de Bruijn graph $G(3)$. Starting from 000, its corresponding de Bruijn sequence is 10111000.

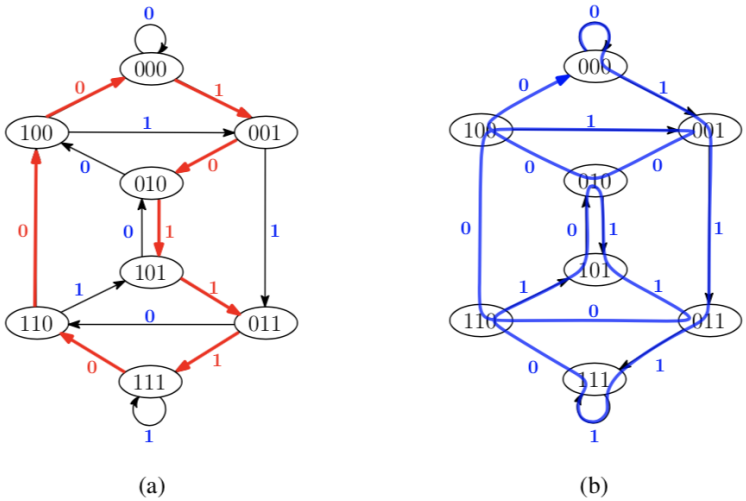


Figure 1 (a) A Hamilton cycle in $G(3)$ starting from 000 corresponding to the de Bruijn sequence 10111000 of order 3. (b) An Euler cycle in $G(3)$ starting from 000 corresponding to the de Bruijn sequence 0111101011001000 of order 4.

Each de Bruijn graph is connected and the in-degree and the out-degree of each vertex is two; the graph $G(n)$ is Eulerian. $G(n)$ is the line graph of $G(n-1)$ which means an Euler cycle in $G(n-1)$ corresponds to a Hamilton cycle in $G(n)$. Thus, the sequence of edge labels visited in an Euler cycle is a de Bruijn sequence. Figure 1(b) illustrates an Euler cycle in $G(3)$. The corresponding de Bruijn sequence of order four when starting from the vertex 000 is 0111101011001000.

Finding an Euler cycle in an Eulerian graph is linear-time solvable with respect to the size of the graph. However, since the graph must be stored, applying such an algorithm to find a de Bruijn sequence requires $O(2^n)$ space. One of the most well-known Euler cycle algorithms for directed graphs is the following due to Fleury [12] with details in [15]. The basic idea is to not burn bridges; in other words, do not visit (and use up) an edge if it leaves the remaining graph disconnected.

Fleury's Euler cycle algorithm (do not burn bridges)

1. Pick a root vertex and compute a spanning in-tree T
2. Make each edge of T (the bridges) the last edge on the adjacency list of the corresponding vertex
3. Starting from the root, traverse edges in a depth-first manner by visiting the first unused edge in the current vertex's adjacency list

Finding a spanning in-tree T can be done by reversing the direction of the edges in the Eulerian graph and computing a spanning out-tree with a standard depth first search on the resulting graph. The corresponding edges in the original graph will be a spanning in-tree. Using this approach, all de Bruijn sequences can be generated by considering all possible spanning in-trees (see BEST Theorem in [15]).

Although not well documented, this algorithm is the basis for all greedy de Bruijn sequence constructions along with their generalizations using preference tables [2] or look-up tables [33]. Specifically, a preference table specifies the precise order that the edges are visited for each vertex when performing Step 3 in Fleury's Euler cycle algorithm. Thus given a preference table and a root vertex, Step 3 in the algorithm can be applied to construct a de Bruijn sequence if combining the last edge from each non-root vertex forms a spanning in-tree to the root. For example, the preference tables and corresponding spanning in-trees for the Prefer-1 (rooted at 000), the Prefer-same (rooted at 010), and the Prefer-opposite (rooted at 000) constructions are given in Figure 2 for $G(3)$. For the Prefer-1, the only valid root is 000. For the Prefer-same, either 010 or 101 could be chosen as root. The Prefer-opposite has a small nuance. By a strict greedy definition, the edges will not create a spanning in-tree for any root. But by changing the preference for the single string 111, a spanning in-tree is created when rooted at 000. This accounts for the special case required in the Prefer-opposite algorithm. Notice how these strings relate to the seeds in their respective greedy constructions. For the Prefer-same, a root of 101 could also have been chosen, and doing so will yield the complement of the Prefer-same sequence when applying this Euler cycle algorithm. Relationships between various preference related constructions have recently been studied in [25], generalizing the work in [29] which focused on the Prefer-opposite and Prefer-1 constructions.

Prefer-1

Vertex	Edge pref
000	1,0
010	1,0
100	1,0
110	1,0
001	1,0
011	1,0
101	1,0
111	1,0

Prefer-same

Vertex	Edge pref
000	0,1
010	0,1
100	0,1
110	0,1
001	1,0
011	1,0
101	1,0
111	1,0

Prefer-opposite

Vertex	Edge pref
000	1,0
010	1,0
100	1,0
110	1,0
001	0,1
011	0,1
101	0,1
111	**1,0**

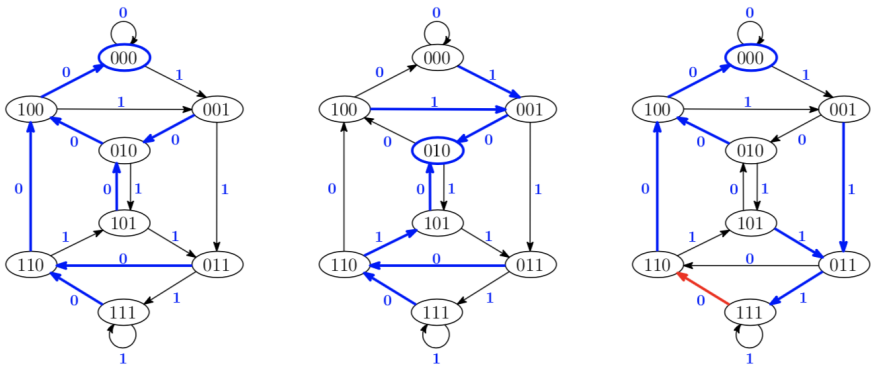


Figure 2 (a) A preference table corresponding to the Prefer-1 greedy construction along with its corresponding spanning in-tree rooted at 000. (b) A preference table corresponding to the Prefer-same greedy construction along with its corresponding spanning in-tree rooted at 010. (c) A preference table corresponding to the Prefer-opposite greedy construction along with its corresponding spanning in-tree rooted at 000.

A second well-known Euler cycle algorithm for directed graphs, attributed to Hierholzer [23], is

as follows:

Hierholzer's Euler cycle algorithm (cycle joining)

1. Start at an arbitrary vertex v visiting edges in a depth-first manner until returning to v, creating a cycle.
2. **Repeat until all edges are visited:** Start from any vertex u on the current cycle and visit remaining edges in a DFS manner until returning to u, creating a new cycle. Join the two cycles together.

This cycle-joining approach is the basis for all successor-rule constructions of de Bruijn sequences. A general framework for joining smaller cycles together based on an underlying feedback shift register is given for the binary case in [20], and then more generally for larger alphabets in [21]. It is the basis for the efficient algorithm presented in this paper, where the initial cycles are induced by a specific feedback function.

3 Run-length encoding

The sequences $\mathcal{S}_n$ and $\mathcal{O}_n$ both have properties based on a run-length encoding of binary strings. The *run-length encoding* (RLE) of a string $\omega = w_1 w_2 \cdots w_n$ is a compressed representation that stores consecutively the lengths of the maximal runs of each symbol. The *run length* of ω is the length of its RLE. For example, the string 11000110 has RLE 2321 and run length 4. Note that 00111001 also has RLE 2321. Since we are dealing with binary strings, we require knowledge of the starting symbol to obtain a given binary string from its RLE. As a further example:

$$\mathcal{S}_5 = 11111000001110110011010001001010 \text{ has RLE } 5531222113121111.$$

The following facts are proved in [3].

▶ **Proposition 1.** *The sequence $\mathcal{S}_n$ is the de Bruijn sequence of order n starting with 1 that has the lexicographically largest RLE.*

▶ **Proposition 2.** *The sequence $\mathcal{O}_n$ is the de Bruijn sequence of order n starting with 1 that has the lexicographically smallest RLE.*

Let $alt(n)$ denote the alternating sequence of 0s and 1s of length n that ends with 0: For example, $alt(6) = 101010$. The following facts are also immediate from [3].

▶ **Proposition 3.** *$\mathcal{S}_n$ has prefix 1^n and has suffix $alt(n-1)$.*

▶ **Proposition 4.** *$\mathcal{O}_n$ has length n prefix $010101\cdots$ and has suffix 10^{n-1}.*

The sequence based on lexicographic compositions [16] also has run-length properties: it is constructed by concatenating lexicographic compositions which are represented using a RLE. Further discussion of this sequence is provided in Section 8.

4 Feedback functions and de Bruijn successors

Let $\mathbf{B}(n)$ denote the set of all binary strings of length n. We call a function $f : \mathbf{B}(n) \to \{0,1\}$ a *feedback function*. Let $\omega = w_1 w_2 \cdots w_n$ be a string in $\mathbf{B}(n)$. A *feedback shift register* is a function $F : \mathbf{B}(n) \to \mathbf{B}(n)$ that takes the form $F(\omega) = w_2 w_3 \cdots w_n f(w_1 w_2 \cdots w_n)$ for a given feedback function f.

A feedback function $g : \mathbf{B}(n) \to \{0,1\}$ is a *de Bruijn successor* if there exists a de Bruijn sequence of order n such that each substring $\omega \in \mathbf{B}(n)$ is followed by $g(\omega)$ in the given de Bruijn

sequence. Given a de Bruijn successor g and a seed string $\omega = w_1 w_2 \cdots w_n$, the following function $\mathrm{DB}(g, \omega)$ will return a de Bruijn sequence of order n with suffix ω:

```
1: function DB(g, ω)
2:     for i ← 1 to 2^n do
3:         x_i ← g(ω)
4:         ω ← w_2 w_3 ··· w_n x_i
5:     return x_1 x_2 ··· x_{2^n}
```

A *linearized de Bruijn sequence* is a linear string that contains every string in $\mathbf{B}(n)$ as a substring exactly once. Such a string has length $2^n + n - 1$. Note that the length n suffix of a de Bruijn sequence $\mathcal{D}_n = \mathrm{DB}(g, w_1 \cdots w_n)$ is $w_1 \cdots w_n$. Thus, $w_2 \cdots w_n \mathcal{D}_n$ is a linearized de Bruijn sequence.

For each of the upcoming feedback functions, selecting appropriate representatives for the cycles they induce is an important step to developing efficient de Bruijn successors for $\mathcal{S}_n$ and $\mathcal{O}_n$. In particular, consider two representatives for a given cycle based on their RLE.

- *RL-rep*: The string with the lexicographically largest RLE; if there are two such strings, it is the one beginning with 1.
- *RL2-rep*: The string with the lexicographically smallest RLE; if there are two such strings, it is the one beginning with 0.

For our upcoming discussion, define the *period* [2] of a string $\omega = w_1 w_2 \cdots w_n$ to be the smallest integer p such that $\omega = (w_1 \cdots w_p)^j$ for some integer j. If $j > 1$ we say that ω is *periodic*; otherwise, we say it is *aperiodic* (or primitive).

4.1 The pure cycling register (PCR)

The *pure cycling register*, denoted PCR, is the feedback shift register with the feedback function $f(\omega) = w_1$. Thus, $\mathrm{PCR}(w_1 w_2 \cdots w_n) = w_2 \cdots w_n w_1$. It is well-known that the PCR partitions $\mathbf{B}(n)$ into cycles of strings that are equivalent under rotation. The following example illustrates the cycles induced by the PCR for $n = 5$ along with their corresponding RL-reps and RL2-reps.

Example 1 The PCR partitions $\mathbf{B}(5)$ into the following eight cycles $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_8$ where the top string in bold is the RL-rep for the given cycle. The underlined string is the RL2-rep.

$\mathbf{P}_1$	$\mathbf{P}_2$	$\mathbf{P}_3$	$\mathbf{P}_4$	$\mathbf{P}_5$	$\mathbf{P}_6$	$\mathbf{P}_7$	$\mathbf{P}_8$
11010	**00101**	**11110**	**00001**	**11100**	**00011**	**11111**	**00000**
10101	01010	11101	00010	11001	00110		
01011	10100	11011	00100	10011	01100		
10110	01001	10111	01000	00111	11000		
01101	10010	01111	10000	01110	10001		

The PCR is the underlying feedback function used to construct the Prefer-1 greedy construction corresponding to the lexicographically largest de Bruijn sequence. It has also been applied in some of the simplest and most efficient de Bruijn sequence constructions [8, 20, 31]. In these constructions, the cycle representatives relate to the lexicographically smallest (or largest) strings in each cycle and they can be determined in $O(n)$ time using $O(n)$ space using standard techniques [5, 9]. We also apply these methods to efficiently determine the RL-reps and the RL2-reps.

[2] The notion of a *period* often allows a fractional exponent j, but here it must be an integer.

Clearly 0^n and 1^n are both RL-reps. Consider a string $\omega = w_1w_2\cdots w_n$ in a cycle $\mathbf{P}$ with RLE $r_1r_2\cdots r_\ell$ where $\ell > 1$. If ω is an RL-rep, then $w_1 \neq w_n$ because otherwise $w_nw_1\cdots w_{n-1}$ has a larger RLE than ω. All strings in $\mathbf{P}$ that differ in the first and last bits form an equivalence class under rotation with respect to their RLE. By definition, the RL-rep will be one that is lexicographically largest amongst all its rotations. As noted above, such a test can be performed in $O(n)$ time using $O(n)$ space. There is one special case to consider: when both a string beginning with 0 and its complement beginning with 1 belong to the same cycle. For example, consider 00101101 and 11010010 which both have RLE 211211. Note this RLE has period $p = 3$ and it is maximal amongst its rotations. By definition, the string beginning with 0 is not an RL-rep. It is not difficult to see that such a string occurs precisely when $w_1 = 0$ and p is odd, where p is the period of $r_1r_2\cdots r_\ell$.

▶ **Proposition 5.** *Let $\omega = w_1w_2\cdots w_n$ be a string with RLE $r_1r_2\cdots r_\ell$, where $\ell > 1$, in a cycle $\mathbf{P}$ induced by the PCR. Let p be the period of $r_1r_2\cdots r_\ell$. Then ω is the RL-rep for $\mathbf{P}$ if and only if*

1. *$w_1 \neq w_n$,*
2. *$r_1r_2\cdots r_\ell$ is lexicographically largest amongst all its rotations, and*
3. *either $w_1 = 1$ or p is even.*

Moreover, testing whether or not ω is an RL-rep can be done in $O(n)$ time using $O(n)$ space.

In a similar manner we consider RL2-reps. Again 0^n and 1^n are both clearly RL2-reps. Consider a string $\omega = w_1w_2\cdots w_n$ in a cycle $\mathbf{P}$ with run length greater than one. If ω is an RL2-rep, then $w_1 \neq w_2$ because otherwise $w_2\cdots w_nw_1$ has a smaller RLE than ω. Thus, consider all strings $s_1s_2\cdots s_n$ in a cycle $\mathbf{P}$ such that $s_2 \neq s_1$. One of these strings is the RL2-rep. Now consider all left rotations of these strings taking the form $s_2\cdots s_ns_1$. Notice that a string in the latter set with the smallest RLE will correspond to the RL2-rep after rotating the string back to the right. As noted in the RL-case, the set of rotated strings form an equivalence class under rotation with respect to their RLE, since their first and last bits differ. Again, the same special case arises as with RL-reps: when both a string beginning with 0 and its complement beginning with 1 belong to the same cycle. For example, consider the cycle containing both 10100101 and 01011010. In each string the first two bits differ. The set of all strings in its cycle where the first two bits differ is $\{10100101, 01001011, 10010110, 01011010, 10110100, 01101001\}$. Rotating each string to the left we get the set $\{01001011, 10010110, 00101101, 10110100, 01101001, 11010010\}$. The corresponding RLEs for this latter set are $\{112112, 121121, 211211, 112112, 121121, 211211\}$. In this case there are two strings 0100100 and 10110100 that both have RLE 112112. Rotating these strings back to the right we have 10100101 and 01011010 which both have the lexicographically smallest RLE of 1112111 in their cycle induced by the PCR. By definition, the string beginning with 0 will be the RL2-rep. Thus ω is not an RL2-rep if $w_1 = 1$, p is odd, and $p < \ell$, where p is the period of the RLE $r_1r_2\cdots r_\ell$ for the string $w_2\cdots w_nw_1$.

▶ **Proposition 6.** *Let $\omega = w_1w_2\cdots w_n$ and let $r_1r_2\cdots r_\ell$ be the RLE of $w_2\cdots w_nw_1$, where $\ell > 1$, in a cycle $\mathbf{P}$ induced by the PCR. Let p be the period of $r_1r_2\cdots r_\ell$. Then ω is the RL2-rep for $\mathbf{P}$ if and only if*

1. *$w_1 \neq w_2$,*
2. *$r_1r_2\cdots r_\ell$ is lexicographically smallest amongst all its rotations, and*
3. *either $w_1 = 0$ or p is even or $p = \ell$.*

Moreover, testing whether or not ω is an RL2-rep can be done in $O(n)$ time using $O(n)$ space.

4.2 The complementing cycling register (CCR)

The *complementing cycling register*, denoted CCR, is the FSR with the feedback function $f(\omega) = \overline{w_1}$, where $\overline{w_1}$ denotes the complement w_1. Thus, CCR$(w_1w_2\cdots w_n) = w_2\cdots w_n\overline{w_1}$. A string and its complement will belong to the same cycle induced by the CCR.

Example 2 The CCR partitions $\mathbf{B}(5)$ into the following four cycles $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4$ where the top string in bold is the RL-rep for the given cycle. The underlined string is the RL2-rep.

$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{C}_3$	$\mathbf{C}_4$
10101	**11101**	**11001**	**11111**
01010	11010	10010	11110
	10100	00100	11100
	01000	01001	11000
	10001	10011	10000
	00010	00110	00000
	00101	01101	00001
	01011	11011	00011
	10111	10110	00111
	01110	01100	01111

The CCR has been applied to efficiently construct de Bruijn sequences in variety of ways [11, 20, 24]. An especially efficient construction applies a concatenation scheme to construct a de Bruijn sequence with discrepancy, which is the maximum difference between the number of 0s and 1s in any substring, bounded above by $2n$ [18, 19].

As with the PCR, we discuss how to efficiently determine whether or not a given string is an RL-rep or an RL2-rep for a cycle $\mathbf{C}$ induced by the CCR. Consider a string $\omega = w_1 w_2 \cdots w_n$ in a cycle $\mathbf{C}$. If ω is an RL-rep, then $w_1 = w_n$ because otherwise $\overline{w_n} w_1 \cdots w_{n-1}$, which is also in $\mathbf{C}$, has a larger RLE than ω. All strings in $\mathbf{C}$ that agree in the first and last bits form an equivalence class under rotation with respect to their RLE (that includes strings starting with both 0 and 1 for each RLE). By definition, the RL-rep will be one that is lexicographically largest amongst all its rotations. As noted in the previous subsection, such a test can be performed in $O(n)$ time using $O(n)$ space. There are no special cases to consider here since a string and its complement always belong to the same cycle. Thus, every RL-rep must begin with 1.

▶ **Proposition 7.** *Let $\omega = w_1 w_2 \cdots w_n$ be a string with RLE $r_1 r_2 \cdots r_\ell$ in a cycle $\mathbf{C}$ induced by the CCR. Then ω is the RL-rep for $\mathbf{C}$ if and only if*

1. *$w_1 = w_n = 1$ and*
2. *$r_1 r_2 \cdots r_\ell$ is lexicographically largest amongst all its rotations.*

Moreover, testing whether or not ω is an RL-rep can be done in $O(n)$ time using $O(n)$ space.

In a similar manner we consider RL2-reps. Again, consider a string $\omega = w_1 w_2 \cdots w_n$ in a cycle $\mathbf{C}$. If ω is an RL2-rep, then $w_1 \neq w_2$ because otherwise $w_2 \cdots w_n \overline{w_1}$ has a smaller RLE than ω. Consider all such strings $w_2 \cdots w_n \overline{w_1}$ in a cycle $\mathbf{C}$ such that $w_2 \neq w_1$. As noted in the RL-case, all such strings form an equivalence class under rotation with respect to their RLE. Clearly, such a string that has the lexicographically smallest RLE will be the RL2-rep. There are no special cases to consider here since a string and its complement always belong to the same cycle. Thus, every RL2-rep must begin with 0 and hence $w_2 = 1$.

▶ **Proposition 8.** *Let $\omega = w_1 w_2 \cdots w_n$ be a string with RLE $r_1 r_2 \cdots r_\ell$ in a cycle $\mathbf{C}$ induced by the CCR. Then ω is the RL2-rep for $\mathbf{C}$ if and only if*

1. *$w_1 = 0$ and $w_2 = 1$, and*
2. *$r_1 r_2 \cdots r_\ell$ is lexicographically smallest amongst all its rotations.*

Moreover, testing whether or not ω is an RL2-rep can be done in $O(n)$ time using $O(n)$ space.

4.3 The pure run-length register (PRR)

The feedback function of particular focus in this paper is $f(\omega) = w_1 \oplus w_2 \oplus w_n$. We will demonstrate that FSR based on this feedback function partitions $\mathbf{B}(n)$ into cycles of strings with the same run

length. Because of this property, we call this FSR the *pure run-length register* and denote it by PRR. Thus,

$$\mathrm{PRR}(w_1w_2\cdots w_n) = w_2\cdots w_n(w_1 \oplus w_2 \oplus w_n).$$

This follows the naming of the pure cycling register (PCR) and the pure summing register (PSR), which is based on the feedback function $f(\omega) = w_1 \oplus w_2 \oplus \cdots \oplus w_n$ [22].

Let $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ denote the cycles induced by the PRR on $\mathbf{B}(n)$. The following example illustrates how the cycles induced by the PRR relate to the cycles induced by the PCR and CCR.

Example 3 The PRR partitions $\mathbf{B}(6)$ into the following 12 cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{12}$ where the top string in bold is the RL-rep for the given cycle. The underlined string is the RL2-rep. The cycles are ordered in non-increasing order with respect to the run lengths of their RL-reps.

$\mathbf{R}_1$	$\mathbf{R}_2$	$\mathbf{R}_3$	$\mathbf{R}_4$	$\mathbf{R}_5$	$\mathbf{R}_6$	$\mathbf{R}_7$	$\mathbf{R}_8$	$\mathbf{R}_9$	$\mathbf{R}_{10}$
101010	**110101**	**001010**	**111010**	**110010**	**111101**	**000010**	**111001**	**000110**	**111110**
010101	101011	010100	110100	100100	111011	000100	110011	001100	111100
	010110	101001	101000	001001	110111	001000	100111	011000	111000
	101101	010010	010001	010011	101111	010000	001110	110001	110000
	011010	100101	100010	100110	011110	100001	011100	100011	100000
			000101	001101					000001
			001011	011011					000011
			010111	110110					000111
			101110	101100					001111
			011101	011001					011111

$\mathbf{R}_{11}$	$\mathbf{R}_{12}$
111111	**000000**

By omitting the last bit of each string, the columns are precisely the cycles of the PCR and CCR for $n = 5$. The cycles $\mathbf{R}_1, \mathbf{R}_4, \mathbf{R}_5, \mathbf{R}_{10}$ relating to the CCR start and end with the different bits. The remaining cycles relate to the PCR; each string in these cycles start and end with the same bit.

In the example above, note that all the strings in a given cycle $\mathbf{R}_i$ have the same run length.

▶ **Lemma 9.** *All the strings in a given cycle $\mathbf{R}_i$ have the same run length.*

Proof. Consider a string $\omega = w_1w_2\cdots w_n$ and the feedback function $f(\omega) = w_1 \oplus w_2 \oplus w_n$. It suffices to show that $w_2\cdots w_nf(\omega)$ has the same run length as ω. This is easily observed since if $w_1 = w_2$ then $w_n = f(\omega)$ and if $w_1 \neq w_2$ then $w_n \neq f(\omega)$. ◀

Based on this lemma, if the strings in $\mathbf{R}_i$ have run length ℓ, we say that $\mathbf{R}_i$ has run length ℓ. Each cycle $\mathbf{R}_i$ has another interesting property: either all the strings start and end with the same bit, or all the strings start and end with different bits. If the strings start and end with the same bit, then $\mathbf{R}_i$ must have odd run length and if we remove the last bit of each string we obtain a cycle induced by the PCR of order $n-1$. In this case we say that $\mathbf{R}_i$ is a *PCR-related cycle*. Such a cycle is *periodic* if for each string $\omega = w_1w_2\cdots w_n \in \mathbf{R}_i$, $w_1w_2\cdots w_{n-1}$ is periodic; otherwise, $\mathbf{R}_i$ is *aperiodic* and the cycle contains $n-1$ distinct strings. If the strings start and end with the different bits, then $\mathbf{R}_i$ must have even run length and if we remove the last bit of each string we obtain a cycle induced by the CCR of order $n-1$. In this case we say that $\mathbf{R}_i$ is a *CCR-related cycle*. Such a cycle is *periodic* if for each string $\omega = w_1w_2\cdots w_n \in \mathbf{R}_i$, $w_1w_2\cdots w_{n-1}\overline{w_1w_2\cdots w_{n-1}}$ is periodic; otherwise, it is *aperiodic* and the cycle contains $2n-2$ distinct strings. As an example, consider the CCR-related cycle for $n = 7$ containing the strings $\{0011001, 0110010, 1100110, 1001101\}$. Consider $\omega = 0011001$ and note that $001100\underline{110011}$ is periodic. These observations were first made in [30] and are illustrated in Example 3, where the periodic cycles are $\mathbf{R}_1$, $\mathbf{R}_{11}$ and $\mathbf{R}_{12}$.

▶ **Observation 10.** *Let $\omega, \omega' \in \mathbf{R}_i$ and let $PRR^j(\omega) = \omega'$. If $\mathbf{R}_i$ is PCR-related then $PRR^{(n-1)-j}(\omega') = \omega$. If $\mathbf{R}_i$ is CCR-related then $PRR^{(2n-2)-j}(\omega') = \omega$ and furthermore $PRR^{n-1}(\omega) = \overline{\omega}$.*

The following lemma considers the RLEs for strings in a cycle $\mathbf{R}_i$.

▶ **Lemma 11.** *Let $\omega = w_1 w_2 \cdots w_n$ be a string in $\mathbf{R}_i$ with RLE of the form $1r_1r_2\cdots r_m$ or $r_1r_2\cdots r_m1$. Then the RLE of any string in $\mathbf{R}_i$ has the form*

$$(r_s{-}j)r_{s+1}\cdots r_m r_1 \cdots r_{s-1}(j{+}1),$$

for some $1 \le s \le m$ and $0 \le j < r_s$.

Proof. If the RLE of ω begins with 1 then $w_1 \neq w_2$ and thus $\mathrm{PRR}(\omega) = w_2 \cdots w_n\overline{w_n}$ will have RLE of the form $r_1r_2\cdots r_m1$. Starting with this RLE, the next $r_1 - 1$ applications of the PRR yield strings with RLE:

$$r_1r_2\cdots r_m1, \quad (r_1{-}1)r_2\cdots r_m2, \quad (r_1{-}2)r_2\cdots r_m3, \quad \ldots, \quad 1r_2\cdots r_m r_1.$$

Repeating this pattern produces the remaining strings in $\mathbf{R}_i$, which leads to the desired result. ◀

Example 4 Consider RL-rep $\omega = 0001110110$ belonging to an aperiodic PCR-related cycle containing the following nine strings with their RLE in parentheses:

0001110110 (33121), 0011101100 (23122), 0111011000 (13123),
1110110001 (31231), 1101100011 (21232), 1011000111 (11233),
0110001110 (12331),
1100011101 (23311), 1000111011 (13312).

We can apply the RL-rep and RL2-rep testers for cycles induced by the PCR and CCR to determine whether or not a string ω is an RL-rep or an RL2-rep for a cycle $\mathbf{R}_i$. These testers, outlined in the following propositions, are critical to the efficiency of our upcoming de Bruijn successors.

▶ **Proposition 12.** *Let $\omega = w_1w_2\cdots w_n$ be a string in a cycle $\mathbf{R}_i$. Then ω is the RL-rep for $\mathbf{R}_i$ if and only if*

1. *$w_1 = w_n$ and $w_1w_2\cdots w_{n-1}$ is an RL-rep with respect to the PCR, or*
2. *$w_1 \neq w_n$ and $w_1w_2\cdots w_{n-1}$ is an RL-rep with respect to the CCR.*

Moreover, testing whether or not ω is an RL-rep for $\mathbf{R}_i$ can be done in $O(n)$ time using $O(n)$ space.

▶ **Proposition 13.** *Let $\omega = w_1w_2\cdots w_n$ be a string in a cycle $\mathbf{R}_i$. Then ω is the RL2-rep for $\mathbf{R}_i$ if and only if*

1. *$w_1 = w_n$ and $w_1w_2\cdots w_{n-1}$ is an RL2-rep with respect to the PCR, or*
2. *$w_1 \neq w_n$ and $w_1w_2\cdots w_{n-1}$ is an RL2-rep with respect to the CCR.*

Moreover, testing whether or not ω is an RL2-rep for $\mathbf{R}_i$ can be done in $O(n)$ time using $O(n)$ space.

The above propositions can easily be verified by the reader based on the definitions of RL-reps and RL2-reps and applying Lemma 11.

5 Generic de Bruijn successors based on the PRR

In this section we provide two generic de Bruijn successors that are applied to derive specific de Bruijn successors for $\mathcal{S}_n$ and $\mathcal{O}_n$ in the subsequent sections. The results relate specifically to the PRR and we assume that $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ denote the cycles induced by the PRR on $\mathbf{B}(n)$.

Let $\omega = w_1 w_2 \cdots w_n$ be a binary string. Define the *conjugate* of ω to be $\hat{\omega} = \overline{w_1} w_2 \cdots w_n$. Similar to Hierholzer's cycle-joining approach discussed in Section 2, Theorem 3.5 from [20] can be applied to systematically join together the *ordered* cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ given certain representatives α_i for each $\mathbf{R}_i$. This theorem is restated as follows when applied to the PRR and the function $f(\omega) = w_1 \oplus w_2 \oplus w_n$.

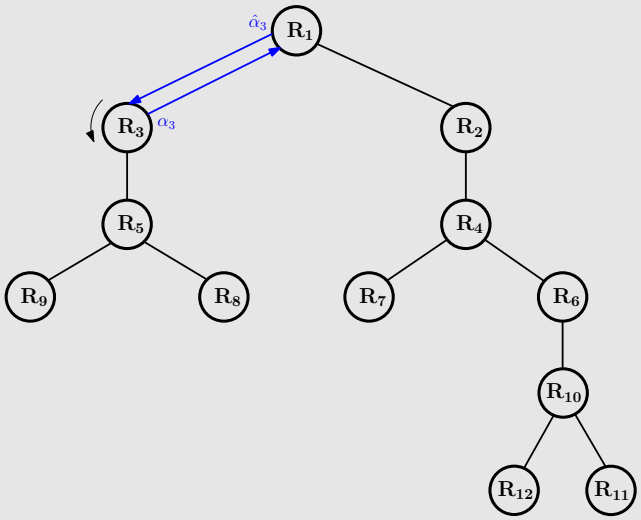
▶ **Theorem 14.** *For each $1 < i \leq t$, if the conjugate $\hat{\alpha}_i$ of the representative α_i for cycle $\mathbf{R}_i$ belongs to some $\mathbf{R}_j$ where $j < i$, then*

$$g(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \{\alpha_2, \alpha_3, \ldots, \alpha_t\}; \\ f(\omega) & \text{otherwise} \end{cases}$$

is a de Bruijn successor.

Together, the ordering of the cycles and the sequence $\alpha_2, \alpha_3, \ldots, \alpha_t$ correspond to a rooted tree, where the nodes are the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ with $\mathbf{R}_1$ designated as the root. There is an edge between two nodes $\mathbf{R}_i$ and $\mathbf{R}_j$ where $i > j$, if and only if $\hat{\alpha}_i$ is in $\mathbf{R}_j$; we say that $\mathbf{R}_j$ is the *parent* of $\mathbf{R}_i$. Each edge represents the joining of two cycles similar to the technique used in Hierholzer's Euler cycle algorithm (see Section 2). An example of such a tree for $n = 6$ is given in the following example.

Example 5 Consider the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{12}$ for $n = 6$ from Example 3 along with their corresponding RL-reps α_i for each $\mathbf{R}_i$. For each $i > 1$, $\hat{\alpha}_i$ belongs to some $\mathbf{R}_j$ where $j < i$. Thus, we can apply Theorem 14 to obtain a de Bruijn successor $g(\omega)$ based on these representatives. The following tree illustrates the joining of these cycles based on g:



Starting with 101010 from $\mathbf{R}_1$, and repeatedly applying the function $g(\omega)$ we obtain the de Bruijn sequence:

$$10101001001100011011001110010101101000100001011101111000000011111.$$

Note that the RL-rep of $\mathbf{R}_3$ is $\alpha_3 = 001010$ and its conjugate $\hat{\alpha}_3 = 101010$ is found in its parent $\mathbf{R}_1$. The last string visited in each cycle $\mathbf{R}_i$, for $i > 1$, is its representative α_i.

The following observations, which will be applied later in our more technical proofs, follow from the tree interpretation of the ordered cycles rooted at $\mathbf{R}_1$ from Theorem 14 as illustrated in the previous

example.[3]

▶ **Observation 15.** *Let g be a de Bruijn successor from Theorem 14 based on representatives $\alpha_2, \alpha_3, \ldots, \alpha_t$. Let $\mathcal{D}_n = DB(g, w_1 w_2 \cdots w_n)$ and let ${\mathcal{D}'}_n = w_2 \cdots w_n \mathcal{D}_n$ denote a linearized de Bruijn sequence. If the length n prefix of ${\mathcal{D}'}_n$ is in $\mathbf{R}_1$, then for each $1 < i \leq t$:*

1. *$\hat{\alpha}_i$ appears before all strings in $\mathbf{R}_i$,*
2. *the m strings of $\mathbf{R}_i$ appear in the following order: $PRR(\alpha_i), PRR^2(\alpha_i), \ldots, PRR^m(\alpha_i) = \alpha_i$,*
3. *if $\mathbf{R}_i$ and $\mathbf{R}_k$ are on the same level in the corresponding tree of cycles rooted at $\mathbf{R}_1$, then either every string in $\mathbf{R}_i$ comes before every string in $\mathbf{R}_k$ or vice-versa,*
4. *the strings in all descendant cycles of $\mathbf{R}_i$ appear after $\hat{\alpha}_i$ and before α_i, and*
5. *if $\hat{\alpha}_i = a_1 a_2 \cdots a_n$, then $a_2 \cdots a_n g(\hat{\alpha}_i)$ is in $\mathbf{R}_i$.*

As an application of Theorem 14, consider the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ to be ordered in *non-increasing order* based on the run length of each cycle. Such an ordering is given in Example 3 for $n = 6$. Using this ordering, let $\alpha_i = a_1 a_2 \cdots a_n$ be any string in $\mathbf{R}_i$, for $i > 1$, such that $a_1 = a_2$. Note that $\hat{\alpha}_i$ has run length that is one *more* than the run length of α_i and thus $\hat{\alpha}_i$ belongs to some $\mathbf{R}_j$ where $j < i$. Thus, Theorem 14 can be applied to describe the following generic de Bruijn successor based on the PRR.

▶ **Theorem 16.** *Let $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ be listed in non-increasing order with respect to the run length of each cycle. Let $\alpha_i = a_1 a_2 \cdots a_n$ denote a representative in $\mathbf{R}_i$ such that $a_1 = a_2$, for each $1 < i \leq t$. Let $\omega = w_1 w_2 \cdots w_n$ and let $f(\omega) = w_1 \oplus w_2 \oplus w_n$. Then the function:*

$$g(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \{\alpha_2, \alpha_3, \ldots, \alpha_t\}; \\ f(\omega) & \text{otherwise.} \end{cases}$$

is a de Bruijn successor.

Now consider the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ to be ordered in *non-decreasing order* based on the run length of each cycle. This means the first two cycles $\mathbf{R}_1$ and $\mathbf{R}_2$ will be the cycles containing 0^n and 1^n. But given this ordering, there is no way to satisfy Theorem 14 since the conjugate of any representative for $\mathbf{R}_2$ will not be found in $\mathbf{R}_1$. However, if we let $\mathbf{R}_t = \{1^n\}$, and order the remaining cycles in *non-decreasing order* based on the run length of each cycle, then we obtain a result similar to Theorem 16. Observe, that this relates to the special case described for the Prefer-opposite greedy construction illustrated in Figure 2. Using this ordering, let $\alpha_i = a_1 a_2 \cdots a_n$ be any string in $\mathbf{R}_i$, for $1 < i < t$, such that $a_1 \neq a_2$. Such a string exists since $\mathbf{R}_1 = \{0^n\}$ and $\mathbf{R}_t = \{1^n\}$. This means $\hat{\alpha}_i$ has run length that is one *less* than the run length of α_i and thus $\hat{\alpha}_i$ belongs to some $\mathbf{R}_j$ where $j < i$. For the special case when $i = t$, the conjugate of 1^n clearly is found in some $\mathbf{R}_j$ where $j < t$. Thus, Theorem 14 can be applied again to describe another generic de Bruijn successor based on the PRR.

▶ **Theorem 17.** *Let $\mathbf{R}_t = \{1^n\}$ and let the remaining cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{t-1}$ be listed in non-decreasing order with respect to the run length of each cycle. Let $\alpha_i = a_1 a_2 \cdots a_n$ denote a representative in $\mathbf{R}_i$ such that $a_1 \neq a_2$, for each $1 < i < t$. Let $\omega = w_1 w_2 \cdots w_n$ and let $f(\omega) = w_1 \oplus w_2 \oplus w_n$. Then the function:*

$$g_2(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \{\alpha_2, \alpha_3, \ldots, \alpha_t\}; \\ f(\omega) & \text{otherwise.} \end{cases}$$

is a de Bruijn successor.

[3] Note that Theorem 14 and Observation 15 apply more generally to any non-singular feedback function f.

When Theorem 16 and Theorem 17 are applied naïvely, the resulting de Bruijn successors are not efficient since storing the set $\{\alpha_2, \alpha_3, \ldots, \alpha_t\}$ requires exponential space. However, if a membership tester for the set can be defined efficiently, then there is no need for the set to be stored. Such sets of representatives are presented in the next two sections.

6 A de Bruijn successor for $\mathcal{S}_n$

In this section we define a de Bruijn successor for $\mathcal{S}_n$. Recall the partition $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ of $\mathbf{B}(n)$ induced by the PRR. In addition to the RL-rep, we define a new representative for each cycle, called the LC-rep, where the LC stands for Lexicographic Compositions which are further discussed in Section 8. Then, considering these two representatives along with a small set of special strings, we define a third representative, called the same-rep. For each representative, we can apply Theorem 16 to produce a new de Bruijn successor. The definitions for these three representatives are as follows:

- *RL-rep*: The string with the lexicographically largest RLE; if there are two such strings, it is the one beginning with 1.
- *LC-rep*: The RL-rep for cycles with run length 1 and n. For all other classes, it is the string ω with RLE $21^{i-1}r_{i+1}\cdots r_\ell$ where $i = \ell$ or $r_{i+1} \neq 1$ such that $\mathrm{PRR}^{i+1}(\omega)$ is the RL-rep.
- *same-rep*: $\begin{cases} \text{RL-rep} & \text{if the RL-rep is } \textit{same-special} \\ \text{LC-rep} & \text{otherwise.} \end{cases}$

We say an RL-rep is *same-special* if it belongs to the set $\mathbf{SP}(n)$ defined as follows:

> $\mathbf{SP}(n)$ is the set of length n binary strings that begin and end with 0 and have RLE of the form $(21^{2x})^y1^z$, where $x \geq 0$, $y \geq 2$, and $z \geq 2$.

The RL-reps have already been illustrated in Section 4. There are relatively few strings in $\mathbf{SP}(n)$ and they all have odd run length since they begin and end with 0; they belong to PCR-related cycles. The need for identifying same-special strings is revealed in the proof for the upcoming Proposition 20.

Example 6 The RLE of the strings in $\mathbf{SP}(n)$ for $n = 10, 11, 12, 13$.

$n = 10$: 2221111
$n = 11$: 2222111, 221111111, 211211111
$n = 12$: 2222211, 222111111
$n = 13$: 222211111, 22111111111, 21121111111

To illustrate an LC-rep, consider the string $\omega = \underline{110101}1111011$ with RLE 2111412. The string ω is an LC-rep since $PRR^5(\omega) = 111101110101$ which is an RL-rep with RLE 413$\underline{1111}$. Note that another way to define the LC-rep is as follows: If the RLE of an RL-rep ends with i consecutive 1s, then the corresponding LC-rep is the string ω such that $PRR^{i+1}(\omega)$ is the RL-rep.

Let $\mathbf{RL}(n)$, $\mathbf{LC}(n)$, and $\mathbf{Same}(n)$ denote the sets of all length n RL-reps, LC-reps, and same-reps, respectively, **not including** the representative with run length n. Consider the following feedback functions where $\omega = w_1w_2\cdots w_n$ and $f(\omega) = w_1 \oplus w_2 \oplus w_n$:

$$RL(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \mathbf{RL}(n); \\ f(\omega) & \text{otherwise,} \end{cases}$$

$$LC(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \mathbf{LC}(n); \\ f(\omega) & \text{otherwise,} \end{cases}$$

$$S(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \mathbf{Same}(n); \\ f(\omega) & \text{otherwise.} \end{cases}$$

▶ **Theorem 18.** *The feedback functions $RL(\omega)$, $LC(\omega)$ and $S(\omega)$ are de Bruijn successors.*

Proof. Let the partition $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ of $\mathbf{B}(n)$ induced by the PRR be listed in non-increasing order with respect to the run length of each cycle. Observe that $\mathbf{R}_1$ is the cycle whose strings have run length n, and thus any representative of $\mathbf{R}_1$ will have run length n. By definition, this representative is not in the sets $\mathbf{RL}(n)$, $\mathbf{LC}(n)$, and $\mathbf{Same}(n)$. Now consider $\mathbf{R}_i$ for $i > 1$. Clearly the RL-rep for $\mathbf{R}_i$ will begin with 00 or 11 and by definition, the LC-rep for $\mathbf{R}_i$ also begins with 00 or 11. Together these results imply that each same-rep for $\mathbf{R}_i$ will also begin with 00 or 11. Thus, it follows directly from Theorem 16 that $RL(\omega)$, $LC(\omega)$ and $S(\omega)$ are de Bruijn successors. ◀

Recall that $alt(n)$ denotes the alternating sequence of 0s and 1s of length n that ends with 0. Let $\mathcal{X}_n = x_1 x_2 \cdots x_{2^n}$ be the de Bruijn sequence returned by $\mathrm{DB}(S, 0alt(n-1))$; it will have suffix equal to the seed $0alt(n-1)$. Let $\mathcal{X}'_n$ denote the linearized de Bruijn sequence $alt(n-1)\mathcal{X}_n$. Our goal is to show that $\mathcal{X}_n = \mathcal{S}_n$. Our proof applies the following two propositions.

▶ **Proposition 19.** *$\mathcal{X}_n$ has prefix 1^n.*

Proof. The result follows from n applications of the successor S to the seed $0alt(n-1)$. ◀

▶ **Proposition 20.** *If β is a string in $\mathbf{B}(n)$ such that the run length of β is one more than the run length of $\hat{\beta}$ and neither β nor $\hat{\beta}$ are same-reps, then $\hat{\beta}$ appears before β in $\mathcal{X}'_n$.*

A proof of this proposition is given later in Section 10.

▶ **Theorem 21.** *The de Bruijn sequences $\mathcal{S}_n$ and $\mathcal{X}_n$ are the same.*

Proof. Let $\mathcal{S}_n = s_1 s_2 \cdots s_{2^n}$, let $\mathcal{X}_n = x_1 x_2 \cdots x_{2^n}$. Recall that $\mathcal{X}_n$ ends with $alt(n-1)$. From Proposition 3 and Proposition 19, $x_1 x_2 \cdots x_n = s_1 s_2 \cdots s_n = 1^n$ and moreover $\mathcal{S}_n$ and $\mathcal{X}_n$ share the same length $n-1$ suffix. Suppose there exists some smallest t, where $n < t \leq 2^n$, such that $s_t \neq x_t$. Let $\beta = x_{t-n} \cdots x_{t-1}$ denote the length n substring of $\mathcal{X}_n$ ending at position $t-1$. Then $x_t \neq x_{t-1}$, because otherwise the RLE of $\mathcal{X}_n$ is lexicographically larger than that of $\mathcal{S}_n$, contradicting Proposition 1. We claim that $\hat{\beta}$ comes before β in $\mathcal{X}'_n$, by considering two cases, recalling $f(\omega) = w_1 \oplus w_2 \oplus w_n$:

- If $x_t = f(\beta)$, then by the definition of S, neither β nor $\hat{\beta}$ are in $\mathbf{Same}(n)$. By the definition of f and since $x_t \neq x_{t-1}$, the first two bits of β must differ from each other. Thus, the run length of β is one more than the run length of $\hat{\beta}$. Thus the claim holds by Proposition 20.
- If $x_t \neq f(\beta)$, then either β or $\hat{\beta}$ are in $\mathbf{Same}(n)$. Let $\beta = b_1 b_2 \cdots b_n$. Then $\mathrm{PRR}(\beta) = b_2 \cdots b_n s_t$ and $\mathrm{PRR}(\hat{\beta}) = b_2 \cdots b_n x_t$. Since $f(\beta) = b_n$, $b_1 = b_2$, which implies $\hat{\beta}$ is not in $\mathbf{Same}(n)$. Thus β is a same-rep and the claim thus holds by Observation 15 (item 1).

Since $\hat{\beta}$ appears before β in $\mathcal{X}'_n$ then $\hat{\beta}$ must be a substring of $alt(n-1)x_1 \cdots x_{t-2}$. Thus, either $x_{t-n+1} \cdots x_{t-1} x_t$ or $x_{t-n+1} \cdots x_{t-1} s_t$ must be in $alt(n-1)x_1 \cdots x_{t-1}$ which contradicts the fact that both $\mathcal{X}_n$ and $\mathcal{S}_n$ are de Bruijn sequences. Thus, there is no $n < t \leq 2^n$ such that $s_t \neq x_t$ and hence $\mathcal{S}_n = \mathcal{X}_n$. ◀

7 A de Bruijn successor for $\mathcal{O}_n$

To develop an efficient de Bruijn successor for $\mathcal{O}_n$, we follow an approach similar to that for $\mathcal{S}_n$, except this time we focus on the lexicographically smallest RLEs and RL2-reps. Again, we consider three different representatives for the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ of $\mathbf{B}(n)$ induced by the PRR.

- *RL2-rep*: The string with the lexicographically smallest RLE; if there are two such strings, it is the one beginning with 0.
- *LC2-rep*: The strings 0^n and 1^n for the classes $\{0^n\}$ and $\{1^n\}$ respectively. For all other classes, it is the string ω with RLE $r_1 r_2 \cdots r_\ell$ such that $r_1 = 1$ and $\text{PRR}^{r_2}(\omega)$ is the RL2-rep.
- *opp-rep*: $\begin{cases} \text{RL2-rep} & \text{if the RL2-rep is } \textit{opp-special} \\ \text{LC2-rep} & \text{otherwise.} \end{cases}$

We say an RL2-rep is *opp-special* if it belongs to the set $\mathbf{SP2}(n)$ defined as follows:

> $\mathbf{SP2}(n)$ is the set of length n binary strings that begin with 1 and have RLE of the form $1x^z y$ where z is odd and $y > x$.

The RL2-reps have already been illustrated in Section 4. There are relatively few strings in $\mathbf{SP2}(n)$ and they all have odd run length; they belong to PCR-related cycles. The need for identifying opp-special strings is revealed in the proof for the upcoming Proposition 24.

Example 7 The RLEs of the strings in $\mathbf{SP2}(n)$ for $n = 10, 11, 12, 13$:

$n = 10$: 111111112, 1111114, 11116, 118, 12223, 127, 136, 145
$n = 11$: 111111113, 1111115, 11117, 119, 12224, 128, 137, 146
$n = 12$: 11111111112, 111111114, 1111116, 11118,11(10), 12225, 129, 138, 147, 156
$n = 13$: 11111111113, 111111115, 1111117, 11119,11(11), 12226, 12(10), 139, 148, 157

Except for the cases 0^n and 1^n, the LC-rep will begin with 10 and 01. As an example, consider $\omega = 10000101001$ which has RLE $r_1 r_2 r_3 r_4 r_5 r_6 r_7 = 1411121$. It is an LC-rep since $\text{PRR}^4(\omega)$ is the RL2-rep 01010010000 with RLE 1111214. Note the last value of this RLE will correspond to r_2.

Let $\mathbf{RL2}(n)$, $\mathbf{LC2}(n)$, and $\mathbf{OPP}(n)$ denote the set of all length n RL2-reps, LC2-reps, and opp-reps, respectively, **not including** the representative 0^n. Consider the following feedback functions where $\omega = w_1 w_2 \cdots w_n$ and $f(\omega) = w_1 \oplus w_2 \oplus w_n$:

$$RL2(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \mathbf{RL2}(n); \\ f(\omega) & \text{otherwise,} \end{cases}$$

$$LC2(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \mathbf{LC2}(n); \\ f(\omega) & \text{otherwise,} \end{cases}$$

$$O(\omega) = \begin{cases} \overline{f(\omega)} & \text{if } \omega \text{ or } \hat{\omega} \text{ is in } \mathbf{OPP}(n); \\ f(\omega) & \text{otherwise.} \end{cases}$$

▶ **Theorem 22.** *The feedback functions $RL2(\omega)$, $LC2(\omega)$ and $O(\omega)$ are de Bruijn successors.*

Proof. Let the partition $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ of $\mathbf{B}(n)$ induced by the PRR be listed such that $\mathbf{R}_t = \{1^n\}$ and the remaining $t-1$ cycles are ordered in non-decreasing order with respect to the run length of each cycle. This means that $\mathbf{R}_1 = \{0^n\}$ and its representative, which must be 0^n, is not in the sets $\mathbf{RL2}(n)$, $\mathbf{LC2}(n)$, and $\mathbf{OPP}(n)$ by their definition. Now consider $\mathbf{R}_i$ for $1 < i < t$. Clearly the RL2-rep for $\mathbf{R}_i$, which is a string with the lexicographically smallest RLE, will begin with 01 or 10. Similarly, the LC2-rep for $\mathbf{R}_i$ must begin with 01 or 10 by its definition. Together these results imply that each opp-rep for $\mathbf{R}_i$ will also begin with 01 or 10. Thus, if follows directly from Theorem 17 that $RL2(\omega)$, $LC2(\omega)$ and $O(\omega)$ are de Bruijn successors. ◀

Recall from Proposition 4 that the length n suffix of $\mathcal{O}_n$ is 10^{n-1}. Let $\mathcal{Y}_n = y_1 y_2 \cdots y_{2^n}$ be the de Bruijn sequence returned by $\text{DB}(O, 10^{n-1})$; it will have suffix 10^{n-1}. Let $\mathcal{Y}'_n$ denote the linearized de Bruijn sequence $0^{n-1}\mathcal{Y}_n$. Our goal is to show that $\mathcal{Y}_n = \mathcal{O}_n$. Our proof applies the following two propositions.

▶ **Proposition 23.** *$\mathcal{Y}_n$ has length n prefix $010101\cdots$.*

Proof. The result follows from n applications of the successor O to the seed 10^{n-1}. ◀

▶ **Proposition 24.** *If β is a string in $\mathbf{B}(n)$ such that the run length of β is one less than the run length of $\hat{\beta}$ and neither β nor $\hat{\beta}$ are opp-reps, then $\hat{\beta}$ appears before β in $\mathcal{Y}'_n$.*

A proof of this proposition is given later in Section 11.

▶ **Theorem 25.** *The de Bruijn sequences $\mathcal{O}_n$ and $\mathcal{Y}_n$ are the same.*

Proof. Let $\mathcal{O}_n = o_1o_2\cdots o_{2^n}$, let $\mathcal{Y}_n = y_1y_2\cdots y_{2^n}$. From Proposition 4 and Proposition 23, $y_1y_2\cdots y_n = o_1o_2\cdots o_n = 0101\cdots$ and moreover $\mathcal{O}_n$ and $\mathcal{Y}_n$ share the same length $n-1$ suffix 0^{n-1}. Based on these prefix and suffix conditions and because both $\mathcal{O}_n$ and $\mathcal{Y}_n$ are de Bruijn sequences, clearly the substring 01^{n-1} is followed by a 1 in both sequences. Suppose there exists some smallest t, where $n < t \leq 2^n$, such that $o_t \neq y_t$. Let $\beta = y_{t-n}\cdots y_{t-1}$ denote the length n substring of $\mathcal{Y}_n$ ending at position $t-1$. Then $y_t = y_{t-1}$, because otherwise the RLE of $\mathcal{Y}_n$ is lexicographically smaller than that of $\mathcal{O}_n$, contradicting Proposition 2. We claim that $\hat{\beta}$ comes before β in $\mathcal{Y}'_n$, by considering two cases, recalling $f(\omega) = w_1 \oplus w_2 \oplus w_n$:

- If $y_t = f(\beta)$, then by the definition of O, neither β nor $\hat{\beta}$ are in $\mathbf{OPP}(n)$. By the definition of f and since $y_t = y_{t-1}$, the first two bits of β are the same. Thus, the run length of β is one less than the run length of $\hat{\beta}$. Thus the claim holds by Proposition 24.
- If $y_t \neq f(\beta)$, then either β or $\hat{\beta}$ are in $\mathbf{OPP}(n)$. Let $\beta = b_1b_2\cdots b_n$. Then $\mathrm{PRR}(\beta) = b_2\cdots b_no_t$ and $\mathrm{PRR}(\hat{\beta}) = b_2\cdots b_ny_t$. Since $f(\beta) \neq b_n$, $b_1 \neq b_2$, which implies $\hat{\beta}$ is not in $\mathbf{OPP}(n)$ since the case when $\beta \neq 01^{n-1}$ was already handled. Thus β is an opp-rep and the claim holds by Observation 15 (item 1).

Since $\hat{\beta}$ appears before β in $\mathcal{Y}'_n$ then $\hat{\beta}$ must be a substring of $0^{n-1}y_1\cdots y_{t-2}$. Thus, either $y_{t-n+1}\cdots y_{t-1}y_t$ or $y_{t-n+1}\cdots y_{t-1}o_t$ must be in $0^{n-1}y_1\cdots y_{t-1}$ which contradicts the fact that both $\mathcal{Y}_n$ and $\mathcal{O}_n$ are de Bruijn sequences. Thus, there is no $n < t \leq 2^n$ such that $o_t \neq y_t$ and hence $\mathcal{O}_n = \mathcal{Y}_n$. ◀

8 Lexicographic compositions

As mentioned earlier, Fredricksen and Kessler devised a construction based on lexicographic compositions [16]. Let $\mathcal{L}_n$ denote the de Bruijn sequence of order n that results from this construction. The sequences $\mathcal{S}_n$ and $\mathcal{L}_n$ first differ at $n = 7$ (as noted below), and for $n \geq 7$ they were conjectured to match for a significant prefix [15, 16]:

$$\mathcal{S}_7 = \begin{array}{l}11111110000000111110111100111101000001000011000010111000111001000110111011000100111010110011001011011010011010100001010010010101010,\end{array}$$

$$\mathcal{L}_7 = \begin{array}{l}11111110000000111110111100111101000001000011000010111000111001000110111011000100111010110011001011011010010100110100100101010.\end{array}$$

After discovering the de Bruijn successor for $\mathcal{S}_n$, we observed that the de Bruijn sequence resulting from the de Bruijn successor $LC(\omega)$ corresponded to $\mathcal{L}_n$ for small values of n. Recall that $alt(n)$ denotes the alternating sequence of 0s and 1s of length n that ends with 0. Let $\mathcal{LC}_n$ be the de Bruijn sequence returned by $\mathrm{DB}(LC, 0alt(n-1))$.

▶ **Conjecture 26.** *The de Bruijn sequences $\mathcal{LC}_n$ and $\mathcal{L}_n$ are the same.*

We verified that $\mathcal{LC}_n$ is the same as $\mathcal{L}_n$ for all $n < 30$. However, as the description of the algorithm to construct $\mathcal{L}_n$ is rather detailed [16], we did not attempt to prove this conjecture.

9 Efficient implementation

Given a membership tester for $\mathbf{RL}(n)$, testing whether or not a string is an LC-rep or a same-rep can easily be done in $O(n)$ time and $O(n)$ space. Similarly, given the membership tester for $\mathbf{RL2}(n)$, testing whether or not a string is an LC2-rep or a opp-rep can easily be done in $O(n)$ time and $O(n)$ space. Thus, by applying Proposition 12 and Proposition 13, we can implement each of our six de Bruijn successors in $O(n)$ time using $O(n)$ space.

▶ **Theorem 27.** *The six de Bruijn successors $RL(\omega)$, $LC(\omega)$, $S(\omega)$, $RL2(\omega)$, $LC2(\omega)$ and $O(\omega)$ can be implemented in $O(n)$ time using $O(n)$ space.*

10 Proof of Proposition 20

Recall that $\mathcal{X}_n$ = DB$(S, 0alt(n-1))$ and $\mathcal{X}'_n = alt(n-1)\mathcal{X}_n$. We begin by restating Proposition 20 by reversing the roles of β and $\hat{\beta}$ in the original statement for convenience:

> If β is a string in $\mathbf{B}(n)$ such that the run length of β is one less than the run length of $\hat{\beta}$ and neither β nor $\hat{\beta}$ are same-reps, then β appears before $\hat{\beta}$ in $\mathcal{X}'_n$.

The first step is to further refine the ordering of the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ used in the proof of Theorem 18 to prove that $S(\omega)$ was a de Bruijn successor. In particular, let $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ be the cycles of $\mathbf{B}(n)$ induced by the PRR ordered in non-increasing order with respect to the run lengths of each cycle, *additionally refined so the cycles with the same run lengths are ordered in decreasing order with respect to the RLE of the RL-rep*. If two RL-reps have the same RLE, then the cycle with RL-rep starting with 1 comes first. Let $\sigma_i, \gamma_i, \alpha_i$ denote the RL-rep, LC-rep, and same-rep, respectively, for $\mathbf{R}_i$, where $1 \leq i \leq t$; let R_i denote the RLE of σ_i.

Assume the run length of β is one less than the run length of $\hat{\beta}$ (the RLE of β must begin with a value greater than 1), and neither β nor $\hat{\beta}$ are same-reps. Since each string in $\mathbf{R}_1$ has maximal run length n, $\beta \in \mathbf{R}_i$ for some $1 < i \leq t$ and thus R_i is of the form $r_1 r_2 \cdots r_m 1^v$ where $r_m > 1$. Let $\mathbf{R}_j$ contain $\hat{\alpha}_i$ which means $\mathbf{R}_j$ is the parent of $\mathbf{R}_i$. Let $\mathbf{R}_k$ contain $\hat{\beta}$. In general, we will show that either $j < k$ or $j = k$; see Figure 3. The cases for when $j < k$ are handled in Section 10.1. In the next steps, we will focus on the situations when $j = k$. Through computer experimentation for $n \leq 25$, we verified that $j = k$ only for specific instances of β equal to γ_i, $\overline{\gamma_i}$, σ_i, or $\overline{\sigma_i}$. In our formal proof, we find that $\mathbf{R}_j$ is aperiodic. Thus, by Observation 15 (item 2), we determine the smallest positive integers a and b such that $\text{PRR}^a(\hat{\alpha}_i) = \hat{\beta}$ and $\text{PRR}^b(\hat{\alpha}_i) = \alpha_j$ and demonstrate that $a < b$.

Outline of next steps:

1. $\sigma_i \in \mathbf{SP}(n)$
2. $\sigma_i \notin \mathbf{SP}(n)$
   - Consider $m = 1$
   - Consider $m > 1$
     - Handle the case when $\beta = \overline{\gamma_i}$
     - Consider one RLE possibility for β which includes an instance when $\beta = \overline{\gamma_i}$
     - Consider a second RLE possibility for β which includes instances when $\beta = \sigma_i$, $\beta = \overline{\sigma_i}$, and $\beta = \overline{\gamma_i}$
     - Handle the instances when $\beta = \sigma_i$ or $\beta = \overline{\sigma_i}$

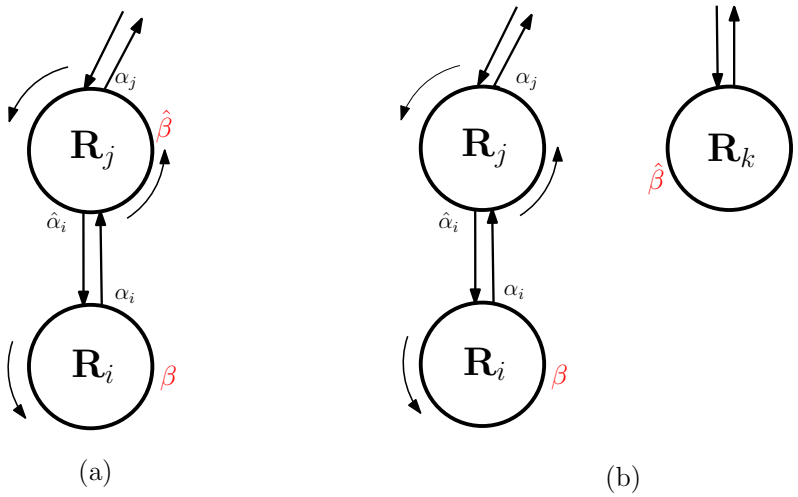


Figure 3 Illustrating the possible relationships between β and $\hat{\beta}$: (a) $j = k$, (b) $j < k$, noting the run length of $\mathbf{R}_j$ and $\mathbf{R}_k$ are the same.

CASE 1: $\sigma_i \in \mathbf{SP}$. In this case, $\alpha_i = \sigma_i$ has RLE of the form $(21^{2x})^y 1^z$ and begins with 0, where $x \geq 0$ and $y, z \geq 2$. Thus $\hat{\alpha}_i \in \mathbf{R}_j$ has RLE $1^{2x+2}(21^{2x})^{y-1}1^z$. Considering the RLE possibilities of the other strings in $\mathbf{R}_j$, as outlined in Lemma 11, we deduce

$$\sigma_j = \text{PRR}^{2x+2}(\hat{\alpha}_i) \text{ begins with 1 and has RLE } (21^{2x})^{y-1}1^{z+2x+2}.$$

Clearly $\mathbf{R}_j$ is aperiodic. Suppose $\beta = \gamma_i$; it will have RLE $21^{z+2x-1}(21^{2x})^{y-1}1$. Observe that $\text{PRR}^{z+2x+1}(\hat{\beta})$ has the same RLE as σ_j, but begins with 0. Thus, since $\mathbf{R}_j$ is CCR-related and applying Observation 10, $\text{PRR}^{(z+2x+1)+(n-1)}(\hat{\beta}) = \sigma_j$, and thus $\text{PRR}^{(n-1)-z+1}(\hat{\alpha}_i) = \hat{\beta}$. By definition, $\text{PRR}^{z+2x+1+2x+2}(\gamma_j) = \sigma_j$, which means that $\text{PRR}^{z+2x+1}(\gamma_j) = \hat{\alpha}_i$. Since $\mathbf{R}_j$ is CCR-related, $\alpha_j = \gamma_j$. Thus $a = n - z$, $b = (2n - 2) - (z + 2x + 1)$, and clearly $a < b$. For all other cases such that $\beta \neq \sigma_i$ (the same-rep), it is a simple exercise to see that $R_j > R_k$, and hence $j < k$.

Example 8 Consider $\mathbf{R}_i$ where $\alpha_i = \sigma_i = 00101101010 \in \mathbf{SP}(n)$ and has RLE 211211111. The corresponding LC-rep $\beta = \gamma_i = 11010100101$ has RLE 211112111. Below are the strings from $\mathbf{R}_j$ including $\hat{\sigma}_i$ and $\hat{\gamma}_i$ in the order that they appear in $\mathcal{X}'_{11}$. Note that $\hat{\beta}$ appears after $\hat{\alpha}_i$ ($a = 8, b = 14$).

01010101011
10101010110
01010101101
10101011010
01010110101
10101101010 ← $\hat{\sigma}_i = \hat{\alpha}_i$
01011010101
10110101010
01101010101
11010101010 ← σ_j, the RL-rep, with RLE 2111111111
10101010100
01010101001
10101010010
01010100101 ← $\hat{\gamma}_i = \hat{\beta}$
10101001010
01010010101
10100101010
01001010101
10010101010
00101010101 ← $\alpha_j = \gamma_j (= \overline{\sigma_j})$, the same-rep and LC-rep for this cycle

CASE 2: $\sigma_i \notin \mathbf{SP}(n)$. By definition $\alpha_i = \gamma_i$. This case involves some rather technical analysis of the RLE for various strings. Assume $R_i = r_1 r_2 \cdots r_m 1^v$, where $m \geq 1$ and $r_1, r_m \geq 2$. Then,

- $\alpha_i = \gamma_i$ has RLE $21^{v-1} r_1 r_2 \cdots r_{m-1}(r_m - 1)$ where $\mathrm{PRR}^{v+1}(\alpha_i) = \sigma_i$, and
- $\hat{\alpha}_i$ has RLE $1^{v+1} r_1 r_2 \cdots r_{m-1}(r_m - 1)$ and is in $\mathbf{R}_j$.

Consider the RLE possibilities of the other strings in $\mathbf{R}_j$ as outlined in Lemma 11. Given that σ_i is an RL-rep, we deduce

$$\sigma_j = \begin{cases} \mathrm{PRR}^{v+1}(\hat{\alpha}_i) & \text{if } \sigma_i \text{ begins with 1;} \\ \mathrm{PRR}^{(n-1)+v+1}(\hat{\alpha}_i) & \text{if } \sigma_i \text{ begins with 0 } (\mathbf{R}_i \text{ is PCR-related).} \end{cases}$$

In both cases σ_j begins with 1 (implying $\alpha_j = \gamma_j$) and $R_j = r_1 r_2 \cdots r_{m-1}(r_m - 1)1^{v+1}$.

▷ Claim 28. If $\mathbf{R}_j$ is the parent of $\mathbf{R}_i$ then σ_j begins with 1 and $\mathbf{R}_j$ is aperiodic.

Note this claim also held for the case when $\sigma_i \in \mathbf{SP}(n)$. Observe that $\mathbf{R}_j$ is indeed aperiodic, since if we assume otherwise, it implies that σ_i is not an RL-rep.

Suppose $m = 1$**.** Then the RLE of β is $r_1 1^v$, the RLE of $\hat{\beta}$ is $1(r_1 - 1)1^v$, the RLE of $\alpha_i = \gamma_i$ is $21^v(r_1 - 1)$ and the RLE of $\hat{\alpha}_i$ is $1^{v+1}(r_1 - 1)$. Thus, $R_j = (r_1 - 1)1^{v+1}$ and the RLE of γ_j is $21^v(r_1 - 2)$. If $\mathbf{R}_i$ is CCR-related and $\beta = \overline{\sigma_i}$, which begins with 0, then $R_j = R_k$ where σ_j begins with 1 and σ_k begins with 0. Thus $j < k$. Otherwise, $\beta = \sigma_i$. By its RLE, $\sigma_j \notin \mathbf{SP}(n)$, so $\alpha_j = \gamma_j$. From the above RLEs, $\mathrm{PRR}^{v+1}(\hat{\alpha}_i) = \hat{\beta}$ and thus $a = v + 1$. Applying Observation 10, if $\mathbf{R}_j$ is a CCR-related cycle and β begins with 1, then it is easily verified that $b = (2n - 2) - 1$ is the smallest value such that $\mathrm{PRR}^b(\hat{\alpha}_i) = \alpha_j$; otherwise $b = (n - 1) - 1$ is the smallest value such that $\mathrm{PRR}^b(\hat{\alpha}_i) = \alpha_j$. In both cases $a < b$.

Suppose $m > 1$**.** Let $d = m$, unless $r_m = 2$, in which case let d be the largest index less than m such that $r_d > 1$. Then given R_j, $\sigma_j = \mathrm{PRR}^{m-d+1+(v+1)}(\gamma_j)$. Thus:

$$\alpha_j = \gamma_j = \begin{cases} \mathrm{PRR}^{(n-1)-(m-d+1)}(\hat{\alpha}_i) & \text{if } \mathbf{R}_i \text{ is CCR-related;} \\ \mathrm{PRR}^{(2n-2)-(m-d+1)}(\hat{\alpha}_i) & \text{if } \mathbf{R}_i \text{ is PCR-related and } \sigma_i \text{ begins with 1;} \\ \mathrm{PRR}^{(n-1)-(m-d+1)}(\hat{\alpha}_i) & \text{if } \mathbf{R}_i \text{ is PCR-related and } \sigma_i \text{ begins with 0.} \end{cases} \quad (1)$$

Consider $\beta = \overline{\gamma_i}$. If $\mathbf{R}_i$ is CCR-related then σ_j begins with 1, but σ_k begins with 0, and hence $j < k$. Otherwise, $\mathbf{R}_i$ is PCR-related and $\mathbf{R}_j$ is CCR-related and hence $\alpha_j = \gamma_j$. Since both γ_i and $\overline{\gamma_i}$ belong to $\mathbf{R}_i$, both σ_i and $\overline{\sigma_i}$ belong to $\mathbf{R}_i$. Thus σ_i begins with 1 and $b = (2n-2) - (m-d+1)$. Since $\alpha_i = \gamma_i$, we have $\mathrm{PRR}^{n-1}(\hat{\alpha}_i) = \overline{\hat{\alpha}_i} = \hat{\beta}$. Thus, $a = n-1$ and clearly $a < b$.

Based on the possibilities given in Lemma 11 using $\omega = \sigma_i$, the RLE for other possible β is of the form

$$(r_s-j)r_{s+1}\cdots r_m 1^{v-1} r_1 \cdots r_{s-1}(j+1),$$

for some $1 \le s \le m$ where $r_s \ge 2$ and $0 \le j \le r_s - 2$. Thus, $\hat{\beta}$ has RLE of the form

$$1(r_s-j-1)r_{s+1}\cdots r_m 1^{v-1} r_1 \cdots r_{s-1}(j+1).$$

Note that $r_1 \ge r_q$ for all $1 < q \le m$. If R_k begins with a value less than r_1, then clearly $R_j > R_k$ and $j < k$. Otherwise, based on the possible RLEs for $\hat{\beta}$ and applying Lemma 11 (using $\omega = \hat{\beta}$), R_k must begin with some $r_{s'} = r_1$ and have the form

$$r_{s'}\cdots r_{s-1}(j+1)(r_s-j-1)r_{s+1}\cdots r_m 1^{v-1} r_1 \cdots r_{s'-1}1, \quad \text{such that } 0 < s' < s, \text{ or} \tag{2}$$
$$r_{s'}\cdots r_m 1^{v-1} r_1 \cdots r_{s-1}(j+1)(r_s-j-1)r_{s+1}\cdots r_{s'-1}1, \quad \text{such that } s < s' \le m, \tag{3}$$

where $0 \le j \le r_s - 2$.

Suppose R_k has the form in (2). Since σ_i is an RL-rep,

$$r_1 \cdots r_m 1^v \;\ge\; r_{s'}\cdots r_{s-1} r_s r_{s+1}\cdots r_m 1^{v-1} r_1 \cdots r_{s'-1}1.$$

We want to compare

$$\begin{aligned} R_j &= r_1 r_2 \cdots r_{m-1}(r_m-1)1^{v+1} \quad \text{with} \\ R_k &= r_{s'}\cdots r_{s-1}(j+1)(r_s-j-1)r_{s+1}\cdots r_m 1^{v-1} r_1 \cdots r_{s'-1}1. \end{aligned}$$

Clearly $R_j > R_k$ unless $r_{s'}\cdots r_{s-1}(j+1) = r_1 \cdots r_{m-1}(r_m-1)$ and $(r_s-j-1)r_{s+1}\cdots r_m 1^{v-1} r_1 \cdots r_{s'-1}1 = 1^{v+1}$. Thus, $s = m$, $s' = 1$, and $j = r_m - 2$. This implies $\beta = \gamma_i$ (same-rep) or $\beta = \overline{\gamma_i}$.

Suppose R_k has the form in (3). Since σ_i is an RL-rep,

$$r_1 \cdots r_m 1^v \;\ge\; r_{s'}\cdots r_m 1^{v-1} r_1 \cdots r_{s-1} r_s r_{s+1}\cdots r_{s'-1}1.$$

We want to compare

$$\begin{aligned} R_j &= r_1 r_2 \cdots r_{m-1}(r_m-1)1^{v+1} \quad \text{with} \\ R_k &= r_{s'}\cdots r_m 1^{v-1} r_1 \cdots r_{s-1}(j+1)(r_s-j-1)r_{s+1}\cdots r_{s'-1}1. \end{aligned}$$

Clearly $R_j \ge R_k$ based on the RLEs described before this claim. Suppose $R_j = R_k$. Then $s' - s \le v$, and the suffix $(r_s-j-1)r_{s+1}\cdots r_{s'-1}1$ must be all 1s with $(j+1) = r_m - 1$. Thus $j = r_s - 2$ and $r_s = r_m$. If $s' - s = v$, then the RLE for β is the same as R_i. Since σ_i is an RL-rep, this implies that $r_1 r_2 \cdots r_m$ is periodic. This is easily deduced since a proper suffix of $r_1 r_2 \cdots r_m$ is equal to a prefix. If $r_1 r_2 \cdots r_m$ is not of the form $(21^p)^q$ for $p \ge 0$ and $q > 1$, then $R_k < R_j$, contradiction. This means $\beta = \gamma_i$ (same-rep), or $\beta = \overline{\gamma_i}$ (already handled). If $s' - s < v$ then clearly $r_s = r_m = 2$, since both $(j+1)$ and (r_s-j-1) must be 1. Suppose $s \ne 1$. Note that $r_{m-s+2}\cdots r_{m-1}(r_m-1)1^{v+1} = r_1 \cdots r_{s-1}11 r_{s+1}\cdots r_{s'-1}1$. However since $s' - s < v$, this implies that $r_1 \cdots r_{s-1} < r_{m-s+2}\cdots r_m$ which contradicts the fact that σ_i is the RL-rep (applying

Lemma 11). Thus $s = 1$ and $r_1 = 2$. Again comparing R_j and R_k:

$$\begin{aligned} R_j &= (r_1 \cdots r_{s'-1})\,(r_{s'} \cdots r_{2s'-1}) \cdots (r_{m-s'} \cdots r_{m-1})1^{v+2} \quad \text{and} \\ R_k &= (r_{s'} \cdots r_{2s'-1}) \cdots (r_{m-s'} \cdots r_{m-1})r_m 1^{v-1}1^{s'+1}. \end{aligned}$$

Since $r_1 \cdots r_{s'-1} = 21^{s'-2}$ and $r_m = 2$, β has RLE of the form $(21^p)^q 1^u$ where: (i) $p \geq 0$ since $p = s' - 2$; (ii) $q > 1$ since $m > 1$; and (iii) $u > 1$ since $u = (v + 2) - (2 + s' - 2) = v - s' + 2$ and $s' - 1 < v$. Thus:

- $\alpha_i = \gamma_i$ has RLE $21^{u+p-1}(21^p)^{q-1}1$,
- $\hat{\alpha}_i$ has RLE $1^{u+p+1}(21^p)^{q-1}1$,
- $\text{PRR}^{u-1}(\hat{\alpha}_i)$ is $\hat{\beta}$ or $\overline{\hat{\beta}}$, and
- $d = m - (p + 1)$ and $m - d + 1 = p + 2$.

Suppose $\mathbf{R}_i$ is PCR-related. Then $\mathbf{R}_j$ is CCR-related cycle and thus $\alpha_j = \gamma_j$. If β begins with 1, then $\beta = \sigma_i$ and $a = u - 1$, and $b = (2n - 2) - (p + 2)$. Clearly $a < b$. If β begins with 0 and p is odd, then $\beta = \sigma_i$, $a = u - 1$, and $b = (n - 1) - (p + 2)$ from (1). Clearly $a < b$. If β begins with 0 and p is even, then if $u > 1$, $\sigma_i \in \mathbf{SP}(n)$, contradiction.

Suppose $\mathbf{R}_i$ is CCR-related. Then $b = (n - 1) - (p + 2)$ from (1). Suppose β begins with 1; $\beta = \sigma_i$. If p is even, then $R_j = R_k$, but σ_j begins with 1 and σ_k begins with 0. Thus $j < k$. If p is odd, then $\sigma_j = \text{PRR}^{p+2}(\hat{\beta})$ and since $v = u + p$, we have $a = u + p + 1 - (p + 2) = u - 1$ and thus $a < b$. Suppose β begins with 0; $\beta = \overline{\sigma_i}$. If p is odd, then $R_j = R_k$, but σ_j begins with 1 and σ_k begins with 0, and hence $j < k$. If p is even, then $\sigma_j = \text{PRR}^{p+2}(\hat{\beta})$ and again $a = u + p + 1 - (p + 2) = u - 1$ and $a < b$.

10.1 $j < k$

The proof for the case when $j < k$ applies the following two claims.

▷ Claim 29. If $\mathbf{R}_j$ and $\mathbf{R}_k$ have the same run length where $j < k$ such that σ_j and σ_k both begin with 1, then every string from $\mathbf{R}_j$ appears in $\mathcal{X}'_n$ before any string from $\mathbf{R}_k$.

Proof. The proof is by induction on the levels of the related tree of cycles rooted by $\mathbf{R}_1$. The base case trivially holds for cycles with run length n since there is only one such cycle $\mathbf{R}_1$. Assume that the result holds for all cycles at levels with run length greater than $\ell < n$. Consider two cycles $\mathbf{R}_j$ and $\mathbf{R}_k$ with run length ℓ such that σ_j and σ_k both begin with 1; neither σ_j nor σ_k are same-special and since $j < k$, $R_j > R_k$. Let $\mathbf{R}_x$ and $\mathbf{R}_y$ denote the parents of $\mathbf{R}_j$ and $\mathbf{R}_k$, respectively. By Claim 28, both σ_x and σ_y begin with 1. Given $R_j > R_k$, our earlier analysis (just before Claim 28) implies that the RLE of σ_x is greater than the RLE of σ_y. Thus, by the ordering of the cycles, $x < y$. By induction, every string from $\mathbf{R}_x$ appears before every string from $\mathbf{R}_y$ in $\mathcal{X}'_n$, and hence by Observation 15 (item 4), we have our result. ◀

▷ Claim 30. Let $\mathbf{R}_k$ and $\mathbf{R}_{k'}$ be cycles with $k' < k$ such that σ_k and $\sigma_{k'}$ have the same RLE $r_1 r_2 \cdots r_m 1^v$ where $r_m > 1$ and $v \geq 0$. Then every string from $\mathbf{R}_{k'}$ appears in $\mathcal{X}'_n$ before any string from $\mathbf{R}_k$.

Proof. By the ordering of the cycles, $\sigma_{k'}$ begins with 1 and σ_k begins with 0; they belong to PCR-related cycles. Note that $\sigma_k = \overline{\sigma_{k'}}$ and similarly $\gamma_k = \overline{\gamma_{k'}}$. Thus, $\hat{\sigma}_k = \overline{\hat{\sigma}_{k'}}$ and $\hat{\gamma}_k = \overline{\hat{\gamma}_{k'}}$ and each pair, respectively, will belong to the same CCR-related cycle. If $\sigma_k \in \mathbf{SP}(n)$, we previously observed that $\hat{\gamma}_k$ and $\hat{\sigma}_k$ belong to the same cycle, and thus $\mathbf{R}_k$ and $\mathbf{R}_{k'}$ have the same parent. If $\sigma_k \notin \mathbf{SP}(n)$, then $\mathbf{R}_k$ and $\mathbf{R}_{k'}$ also have the same parent containing both $\hat{\gamma}_k$ and $\hat{\gamma}_{k'}$. Let $\mathbf{R}_\ell$ be the shared parent of $\mathbf{R}_k$ and $\mathbf{R}_{k'}$. Since $\sigma_{k'}$ begins with 1, $\alpha_{k'} = \gamma_{k'}$. If $m = 1$, then we

already saw that $\alpha_\ell = \text{PRR}^{(2n-3)}(\hat{\alpha}_{k'})$ and $\alpha_\ell = \text{PRR}^{(n-2)}(\hat{\alpha}_k)$. If $m > 1$, we observed that $\alpha_\ell = \text{PRR}^{(2n-2)-(m-d+1)}(\hat{\alpha}_{k'})$ from (1), recalling $d = m$ unless $r_m = 2$, in which case let d is the largest index less than m such that $r_d > 1$. If $\sigma_k \in \mathbf{SP}(n)$ then $\alpha_k = \sigma_k$ and from earlier analysis $\alpha_\ell = \text{PRR}^{(2n-2)-(v+1)}(\hat{\alpha}_k)$ noting $(m-d+1) \leq v$ in this case; otherwise, $\alpha_k = \gamma_k$, and $\alpha_\ell = \text{PRR}^{(n-1)-(m-d+1)}(\hat{\alpha}_k)$ from (1). In all cases, applying Observation 15, $\hat{\alpha}_{k'}$ appears before $\hat{\alpha}_k$ in $\mathcal{X}'_n$, and every string in $\mathbf{R}_{k'}$ appears in $\mathcal{X}'_n$ before any string in $\mathbf{R}_k$. ◀

Recall that σ_j begins with 1 from Claim 28. Thus, if σ_k begins with 1, then Claim 29 implies that all strings from $\mathbf{R}_j$ appear in $\mathcal{X}'_n$ before all strings from $\mathbf{R}_k$. Otherwise, if σ_k begins with 0, then it must correspond to a PCR-related cycle. Consider $\mathbf{R}_{k'}$ containing RL-rep $\overline{\sigma_k}$ which begins with 1; it has the same RLE as σ_k. By Claim 30, all strings from $\mathbf{R}_{k'}$ appear in $\mathcal{X}'_n$ before all strings from $\mathbf{R}_k$. If $j = k'$, we are done; otherwise $j < k' < k$ and Claim 29 implies that all strings from $\mathbf{R}_j$ appear in $\mathcal{X}'_n$ before all strings from $\mathbf{R}_{k'}$. Finally, by applying Observation 15 (item 4), all strings from $\mathbf{R}_i$ including β will appear in $\mathcal{X}'_n$ before all strings from $\mathbf{R}_k$ including $\hat{\beta}$.

11 Proof of Proposition 24

The proof of this proposition follows similar steps as the proof for Proposition 20; however, the RLE analysis is less complex. Recall that $\mathcal{Y}_n = \text{DB}(O, 10^{n-1})$ and $\mathcal{Y}'_n = 0^{n-1}\mathcal{Y}_n$. We restate Proposition 24, reversing the roles of β and $\hat{\beta}$ from its original statement for convenience:

> If β is a string in $\mathbf{B}(n)$ such that the run-length of β is one more than the run-length of $\hat{\beta}$ and neither β nor $\hat{\beta}$ are opp-reps, then β appears before $\hat{\beta}$ in $\mathcal{Y}'_n$.

The first step is to further refine the ordering of the cycles $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_t$ used in the proof of Theorem 22 to prove that $O(\omega)$ was a de Bruijn successor. In particular, let $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{t-1}$ be the cycles of $\mathbf{B}(n)$ induced by the PRR, not including $\mathbf{R}_t = \{1^n\}$, ordered in non-decreasing order with respect to the run lengths of each cycle. *This ordering is additionally refined so the cycles with the same run lengths are ordered in increasing order with respect to the RLE of the RL2-rep*. If two RL2-reps have the same RLE, then the cycle with RL2-rep starting with 0 comes first. Let $\sigma_i, \gamma_i, \alpha_i$ denote the RL2-rep, LC2-rep, and opp-rep, respectively, for $\mathbf{R}_i$, where $1 \leq i \leq t$; let R_i denote the RLE of σ_i. Assume the run length of β is one more than the run length of $\hat{\beta}$, and neither β nor $\hat{\beta}$ are opp-reps. This run-length constraint implies that the RLE of β must begin with 1. Since each string in $\mathbf{R}_1$ and $\mathbf{R}_t$ has run length 1, $\beta \in \mathbf{R}_i$ for some $1 < i < t$. Let $\mathbf{R}_j$ contain $\hat{\alpha}_i$ which means $\mathbf{R}_j$ is the parent of $\mathbf{R}_i$. Let $\mathbf{R}_k$ contain $\hat{\beta}$. Like the proof in the previous section, we show that either $j < k$ or $j = k$; see Figure 3. The cases for when $j < k$ are handled in Section 11.1. As we analyze the cases when $j = k$, we find that $\mathbf{R}_j$ is aperiodic. Thus, by Observation 15 (item 2), we determine the smallest positive integers a and b such that $\text{PRR}^a(\hat{\alpha}_i) = \hat{\beta}$ and $\text{PRR}^b(\hat{\alpha}_i) = \alpha_j$ and demonstrate that $a < b$.

CASE 1: $\sigma_i \in \mathbf{SP2}(n)$. In this case $\alpha_i = \sigma_i$ begins with 1 and $R_i = 1x^z y$ where z is odd and $y > x$. Thus $\hat{\alpha}_i \in \mathbf{R}_j$ begins with 0 and has RLE $(x{+}1)x^{z-1}y$. Considering the RLE possibilities of the other strings in $\mathbf{R}_j$, as outlined in Lemma 11, clearly

$$\sigma_j = \text{PRR}^x(\hat{\alpha}_i) \text{ begins with 0 and has RLE } 1x^{z-1}(y{+}x),$$

and $\mathbf{R}_j$ is aperiodic. Suppose $\beta = \gamma_i$; it will have RLE $1yx^z$ and begin with 0. Observe that $\text{PRR}^y(\hat{\beta})$ has the same RLE as σ_j, but begins with 1. Thus, since $\mathbf{R}_j$ is CCR-related and applying Observation 10, $\text{PRR}^{y+(n-1)}(\hat{\beta}) = \sigma_j$, and thus $\text{PRR}^{(n-1)+x-y}(\hat{\alpha}_i) = \hat{\beta}$. By definition, $\text{PRR}^{y+x}(\gamma_j) = \sigma_j$, which means that $\text{PRR}^y(\gamma_j) = \hat{\alpha}_i$. Since $\mathbf{R}_j$ is CCR-related, $\alpha_j = \gamma_j$. Thus $a = (n{-}1) + x - y$,

$b = (2n-2) - y$, and clearly $a < b$. For all other cases such that $\beta \neq \sigma_i$ (the opp-rep), it is a simple exercise to see that $R_j < R_k$, and hence $j < k$.

Example 9 Consider $\mathbf{R}_i$ where $\alpha_i = \sigma_i = 10011001111 \in \mathbf{SP2}(11)$ and has RLE 12224. The corresponding LC2-rep $\beta = \gamma_i = 01111001100$ has RLE 14222. Below are the strings from $\mathbf{R}_j$ including $\hat{\sigma}_i$ and $\hat{\gamma}_i$ in the order that they appear in $\mathcal{Y}'_{11}$. Note that $\hat{\beta}$ appears after $\hat{\alpha}_i$ ($a = 8, b = 16$).

00000011001
00000110011
00001100111
00011001111 ← $\hat{\sigma}_i = \hat{\alpha}_i$
00110011111
01100111111 ← σ_j, the RL2-rep, with RLE 1226
11001111110
10011111100
00111111001
01111110011
11111100110
11111001100 ← $\hat{\gamma}_i = \hat{\beta}$
11110011000
11100110000
11001100000
10011000000 ← $\overline{\sigma_j}$
00110000001
01100000011
11000000110
10000001100 ← $\alpha_j = \gamma_j$, the opp-rep and LC2-rep for this cycle

CASE 2: $\sigma_i \notin \mathbf{SP2}(n)$. If $m = 1$ then $\mathbf{R}_i$ is CCR-related and $R_i = 1r_1$. Thus $\beta = 01^{n-1} = \sigma_i$ since it is is not an opp-rep. However, $\hat{\beta} = 1^n$ is an opp-rep. Contradiction. Thus, assume $m > 1$. By definition $\alpha_i = \gamma_i$. Assume $R_i = 1r_1r_2 \cdots r_m$. Then,

- $\alpha_i = \gamma_i$ has RLE $1r_mr_1r_2 \cdots r_{m-1}$ where $\text{PRR}^{r_m}(\alpha_i) = \sigma_i$, and
- $\hat{\alpha}_i$ has RLE $(r_m{+}1)r_1r_2 \cdots r_{m-1}$ and is in $\mathbf{R}_j$.

Consider the RLE possibilities of the other strings in $\mathbf{R}_j$ as outlined in Lemma 11. Given that σ_i is an RL-rep, clearly

$$\sigma_j = \begin{cases} \text{PRR}^{r_m}(\hat{\alpha}_i) & \text{if } \sigma_i \text{ begins with 0;} \\ \text{PRR}^{(n-1)+r_m}(\hat{\alpha}_i) & \text{if } \sigma_i \text{ begins with 1 } (\mathbf{R}_i \text{ is PCR-related).} \end{cases}$$

In both cases σ_j begins with 0 (implying $\alpha_j = \gamma_j$) and $R_j = 1r_1r_2 \cdots r_{m-2}(r_{m-1}{+}r_m)$.

▷ Claim 31. If $\mathbf{R}_j$ is the parent of $\mathbf{R}_i$ then σ_j begins with 0 and $\mathbf{R}_j$ is aperiodic.

Note this claim also held for the case when $\sigma_i \in \mathbf{SP2}(n)$. Observe that $\mathbf{R}_j$ is indeed aperiodic, since if we assume otherwise, it implies that σ_i is not an RL2-rep. By definition of an LC2-rep, $\sigma_j = \text{PRR}^{r_{m-1}+r_m}(\gamma_j)$. Thus:

$$\alpha_j = \gamma_j = \begin{cases} \text{PRR}^{(n-1)-r_{m-1}}(\hat{\alpha}_i) & \text{if } \mathbf{R}_i \text{ is CCR-related;} \\ \text{PRR}^{(2n-2)-r_{m-1}}(\hat{\alpha}_i) & \text{if } \mathbf{R}_i \text{ is PCR-related and } \sigma_i \text{ begins with 0;} \\ \text{PRR}^{(n-1)-r_{m-1}}(\hat{\alpha}_i) & \text{if } \mathbf{R}_i \text{ is PCR-related and } \sigma_i \text{ begins with 1.} \end{cases} \quad (4)$$

Suppose $\beta = \overline{\gamma_i}$. If $\mathbf{R}_i$ is CCR-related then σ_j begins with 0, but σ_k begins with 1, and hence $j < k$. Otherwise, $\mathbf{R}_i$ is PCR-related and $\mathbf{R}_j$ is CCR-related and hence $\alpha_j = \gamma_j$. Since both γ_i and $\overline{\gamma_i}$ belong to $\mathbf{R}_i$, both σ_i and $\overline{\sigma_i}$ belong to $\mathbf{R}_i$. Thus σ_i begins with 0 and from (4), $b = (2n-2) - r_{m-1}$. Since $\alpha_i = \gamma_i$, we have $\text{PRR}^{n-1}(\hat{\alpha}_i) = \overline{\hat{\alpha}_i} = \hat{\beta}$. Thus, $a = n-1$ and clearly $a < b$.

Since the RLE of β begins with 1, from Lemma 11, the RLE for β must be of the form $1r_s \cdots r_m r_1 \cdots r_{s-1}$ for some $1 \leq s \leq m$. Similar to our analysis for R_j, R_k must begin with 1 followed by a rotation of $r_{s+1} \cdots r_m r_1 \cdots r_{s-2}(r_{s-1}+r_s)$. Suppose $1 < s \leq m$. Let $r_1 \cdots r_m = (r_1 \cdots r_p)^q$ for some largest $q \geq 1$. Then since σ_i is an RL2-rep, $R_j < R_k$ unless s is a multiple of p, in which case $\beta = \gamma_i$ (opp-rep) or $\beta = \overline{\gamma_i}$ (already handled). Suppose $s = 1$, which means $\beta = \sigma_i$ or $\beta = \overline{\sigma_i}$. Since σ_i is an RL2-rep, for each $1 < s' \leq m$, the string $r_{s'} \cdots r_m$ is less than or equal to the prefix of σ_i of the same length. Thus, for $s' \neq 2$, $R_j < R_k$. If $s' = 2$, $R_j < R_k$ unless $r_1 \cdots r_{m-2} = r_2 \cdots r_{m-1}$ and $r_1 = r_{m-1}$, in which case $R_j = R_k$. Since σ_i is an RL2-rep, $r_m \geq r_1$. Thus, β has RLE of the form $1x^z y$ where $y \leq x$. If $y = x$, then $\beta = \gamma_i$ (opp-rep) or $\beta = \overline{\gamma_i}$ (already handled). Thus consider $y > x$. We now consider whether or not $\mathbf{R}_i$ is CCR-related or PCR-related. Note $r_{m-1} = x$ and $r_m = y$.

Suppose $\mathbf{R}_i$ is CCR-related. If $\beta = \sigma_i$, then σ_j begins with 0, but σ_k begins with 1 and thus $j < k$. Otherwise, if $\beta = \overline{\sigma_i}$ then $j = k$. From (4), $b = (n-1) - x$. Note that $\text{PRR}^x(\hat{\beta}) = \sigma_j$ and previously we observed that $\text{PRR}^y(\hat{\alpha}_i) = \sigma_j$. Thus, $a = y - x$ and clearly $a < b$.

Suppose $\mathbf{R}_i$ is PCR-related. By its RLE, clearly $\overline{\sigma_i}$ is not in $\mathbf{R}_i$. Thus $\beta = \sigma_i$. If β begins with 1, then since z must be odd, $\beta \in \mathbf{SP2}(n)$ – contradiction. If β begins with 0, then observe that $\text{PRR}^x(\hat{\beta}) = \overline{\sigma_j}$ and hence $\text{PRR}^{(n-1)-x}(\sigma_j) = \hat{\beta}$. Thus $a = (n-1) - x + y$. From (4), $b = (2n-2) - x$ and clearly $a < b$.

11.1 $j < k$

This section applies the same arguments as Section 10.1.

▷ Claim 32. If $\mathbf{R}_j$ and $\mathbf{R}_k$ have the same run length where $j < k$ such that σ_j and σ_k both begin with 0, then every string from $\mathbf{R}_j$ appears in $\mathcal{Y}'_n$ before any string from $\mathbf{R}_k$.

Proof. The proof is by induction on the levels of the related tree of cycles rooted by $\mathbf{R}_1$. The base case trivially holds for cycles with run length 1, as there are not two cycles that meet the conditions. Assume that the result holds for all cycles at levels with run length less than $\ell > 1$. Consider two cycles $\mathbf{R}_j$ and $\mathbf{R}_k$ with run length ℓ such that σ_j and σ_k both begin with 0; neither σ_j nor σ_k are opp-special and since $j < k$, $R_j < R_k$. Let $\mathbf{R}_x$ and $\mathbf{R}_y$ denote the parents of $\mathbf{R}_j$ and $\mathbf{R}_k$, respectively. By Claim 31, both σ_x and σ_y begin with 0. Given $R_j > R_k$, our earlier analysis (just before Claim 31) implies that the RLE of σ_x is less than the RLE of σ_y. Thus, by the ordering of the cycles, $x < y$. By induction, every string from $\mathbf{R}_x$ appears before every string from $\mathbf{R}_y$ in $\mathcal{Y}'_n$, and hence by Observation 15 (item 4), we have our result. ◀

▷ Claim 33. Let $\mathbf{R}_k$ and $\mathbf{R}_{k'}$ be cycles with $k' < k$ such that σ_k and $\sigma_{k'}$ have the same RLE $1r_1r_2 \cdots r_m$ where $m \geq 1$. Then every string from $\mathbf{R}_{k'}$ appears in $\mathcal{Y}'_n$ before any string from $\mathbf{R}_k$.

Proof. By the ordering of the cycles, $\sigma_{k'}$ begins with 0 and σ_k begins with 1; they belong to PCR-related cycles. Note that $\sigma_k = \overline{\sigma_{k'}}$ and similarly $\gamma_k = \overline{\gamma_{k'}}$. Thus, $\hat{\sigma}_k = \overline{\hat{\sigma}_{k'}}$ and $\hat{\gamma}_k = \overline{\hat{\gamma}_{k'}}$ and each pair, respectively, will belong to the same CCR-related cycle. If $\sigma_k \in \mathbf{SP2}(n)$ we previously observed that $\hat{\gamma}_k$ and $\hat{\sigma}_k$ belong to the same cycle, and thus $\mathbf{R}_k$ and $\mathbf{R}_{k'}$ have the same parent. If $\sigma_k \notin \mathbf{SP2}(n)$, then $\mathbf{R}_k$ and $\mathbf{R}_{k'}$ also have the same parent containing both $\hat{\gamma}_k$ and $\hat{\gamma}_{k'}$. Let $\mathbf{R}_\ell$ be the shared parent of $\mathbf{R}_k$ and $\mathbf{R}_{k'}$. Since $\sigma_{k'}$ begins with 0, $\alpha_{k'} = \gamma_{k'}$. If $m = 1$ there is only one cycle and it is CCR-related. If $m > 1$, then $\alpha_\ell = \text{PRR}^{(2n-2)-r_{m-1}}(\hat{\alpha}_{k'})$ from (4). If $\sigma_k \in \mathbf{SP2}(n)$, then

$\alpha_k = \sigma_k$ and from earlier analysis $\alpha_\ell = \text{PRR}^{(2n-2)-r_m}(\hat{\alpha}_k)$, where $r_m = y$; otherwise $\alpha_k = \gamma_k$, and $\alpha_\ell = \text{PRR}^{(n-1)-r_{m-1}}(\hat{\alpha}_k)$. In both cases, applying Observation 15, $\hat{\alpha}_{k'}$ appears before $\hat{\alpha}_k$ in $\mathcal{Y}'_n$, and every string in $\mathbf{R}_{k'}$ appears in $\mathcal{Y}'_n$ before any string in $\mathbf{R}_k$. ◀

Recall that σ_j begins with 0 from Claim 31. Thus, if σ_k begins with 0, then Claim 32 implies that all strings from $\mathbf{R}_j$ appear in $\mathcal{Y}'_n$ before all strings from $\mathbf{R}_k$. Otherwise, if σ_k begins with 1, then it must correspond to a PCR-related cycle. Consider $\mathbf{R}_{k'}$ containing RL2-rep $\overline{\sigma_k}$ which begins with 0; it has the same RLE as σ_k. By Claim 33, all strings from $\mathbf{R}_{k'}$ appear in $\mathcal{Y}'_n$ before all strings from $\mathbf{R}_k$. If $j = k'$, we are done; otherwise $j < k' < k$ and Claim 32 implies that all strings from $\mathbf{R}_j$ appear in $\mathcal{Y}'_n$ before all strings from $\mathbf{R}_{k'}$. Finally, by applying Observation 15 (item 4), all strings from $\mathbf{R}_i$ including β will appear in $\mathcal{Y}'_n$ before all strings from $\mathbf{R}_k$ including $\hat{\beta}$.

12 Future work

The following questions provide avenues for future research.

P1. Can $\mathcal{S}_n$, $\mathcal{O}_n$, or $\mathcal{L}_n$ be generated via a concatenation approach, and if so, can they be generated in $O(1)$ time per symbol using polynomial space?

P2. The (greedy) prefer-same and prefer-opposite de Bruijn sequences for alphabets of size $k > 2$ are described at `http://debruijnsequence.org`. Are there simple de Bruijn successors for these generalized sequences?

P3. Does there exist an efficient decoding algorithm for the sequences $\mathcal{S}_n$, $\mathcal{O}_n$, or $\mathcal{L}_n$? That is, without generating the sequence, at what position r do we find a given string ω (unranking)? And, given a string ω, at what position r does it appear (ranking)?

P4. Answer Conjecture 26.

P5. Can Fredricksen and Kessler's de Bruijn sequence construction $\mathcal{L}_n$ [16] be generalized to larger alphabets?

References

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

2 A. Alhakim. Spans of preference functions for de Bruijn sequences. *Discrete Applied Mathematics*, 160(7-8):992 – 998, 2012.

3 A. Alhakim, E. Sala, and J. Sawada. Revisiting the prefer-same and prefer-opposite de Bruijn sequence constructions. *Theoretical Computer Science*, 852:73–77, 2021.

4 J. Aycock. *Retrogame Archeology*. Springer International Publishing, 2016.

5 K. S. Booth. Lexicographically least circular substrings. *Inform. Process. Lett.*, 10(4/5):240–242, 1980.

6 P. E. C. Compeau, P. A. Pevzner, and G. Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11):987–991, 2011.

7 N. G. de Bruijn. A combinatorial problem. *Indagationes Mathematicae*, 8:461–467, 1946.

8 P. B. Dragon, O. I. Hernandez, J. Sawada, A. Williams, and D. Wong. Constructing de Bruijn sequences with co-lexicographic order: the k-ary Grandmama sequence. *European J. Combin.*, 72:1–11, 2018.

9 J. P. Duval. Factorizing words over an ordered alphabet. *Journal of Algorithms*, 4(4):363–381, 1983.

10 C. Eldert, H. Gray, H. Gurk, and M. Rubinoff. Shifting counters. *AIEE Trans.*, 77:70–74, 1958.

11 T. Etzion. Self-dual sequences. *Journal of Combinatorial Theory, Series A*, 44(2):288 – 298, 1987.

12 M. Fleury. Deux problemes de geometrie de situation. *Journal de mathematiques elementaires*, 42:257–261, 1883.

13 C. Flye Sainte-Marie. Solution to question nr. 48. *L'intermédiaire des Mathématiciens*, 1:107–110, 1894.

14 H. Fredricksen. Generation of the Ford sequence of length 2^n, n large. *J. Combin. Theory Ser. A*, 12(1):153–154, 1972.

15 H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *Siam Review*, 24(2):195–221, 1982.

16 H. Fredricksen and I. Kessler. Lexicographic compositions and de Bruijn sequences. *J. Combin. Theory Ser. A*, 22(1):17 – 30, 1977.

17 H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Math.*, 23:207–210, 1978.

18 D. Gabric and J. Sawada. Constructing de Bruijn sequences by concatenating smaller universal cycles. *Theoretical Computer Science*, 743:12 – 22, 2018.

19 D. Gabric and J. Sawada. Investigating the discrepancy property of de Bruijn sequences. *Submitted manuscript*, 2020.

20 D. Gabric, J. Sawada, A. Williams, and D. Wong. A framework for constructing de Bruijn sequences via simple successor rules. *Discrete Mathematics*, 341(11):2977 – 2987, 2018.

21 D. Gabric, J. Sawada, A. Williams, and D. Wong. A successor rule framework for constructing k -ary de Bruijn sequences and universal cycles. *IEEE Transactions on Information Theory*, 66(1):679–687, 2020.

22 S. W. Golomb. *Shift Register Sequences*. Aegean Park Press, Laguna Hills, CA, USA, 1981.

23 C. Hierholzer. Deux problemes de geometrie de situation. *Journal de mathematiques elementaires*, 42:257–261, 1873.

24 Y. Huang. A new algorithm for the generation of binary de Bruijn sequences. *J. Algorithms*, 11(1):44–51, 1990.

25 Y. Jiang. A relation between sequences generated by Golomb's preference algorithm. *Designs, Codes and Cryptography*, 91(1):285–291, Jan 2023.

26 A. Klein. *Stream Ciphers*. Springer-Verlag London, 2013.

27 M. H. Martin. A problem in arrangements. *Bull. Amer. Math. Soc.*, 40(12):859–864, 1934.

28 P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

29 A. Rubin and G. Weiss. Mapping prefer-opposite to prefer-one de Bruijn sequences. *Designs, Codes and Cryptography*, 85(3):547–555, Dec 2017.

30 E. Sala. Exploring the greedy constructions of de Bruijn sequences. Master's thesis, University of Guelph, 2018.

31 J. Sawada, A. Williams, and D. Wong. A surprisingly simple de Bruijn sequence construction. *Discrete Math.*, 339:127–131, 2016.

32 A. Williams. The greedy Gray code algorithm. In F. Dehne, R. Solis-Oba, and J.-R. Sack, editors, *Algorithms and Data Structures*, pages 525–536, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

33 S. Xie. Notes on de Bruijn sequences. *Discrete Applied Mathematics*, 16(2):157 – 177, 1987.

A Implementation of the de Bruijn successors $RL(\omega)$, $LC(\omega)$, and $S(\omega)$

```
#include<stdio.h>
#include<math.h>
#define N_MAX 50
int n;

// ================================================================================
// Compute the RLE of a[1..m] in run[1..r], returning r = ruh length
// ================================================================================
int RLE(int a[], int run[], int m) {
     int i,j,r,old;

     old = a[m+1];
     a[m+1] = 1 - a[m];
     r = j = 0;
     for (i=1; i<=m; i++) {
          if (a[i] == a[i+1]) j++;
          else {  run[++r] = j+1;  j = 0;  }
     }
     a[m+1] = old;
     return r;
}
// ================================================================================
// Check if a[1..n] is a "special" RL representative.  It must be that a[1] = a[n]
// and the RLE of a[1..n] is of the form (21^j)^s1^t where j is even, s >=2, t>=2
// ================================================================================
int Special(int a[]) {
     int i,j,r,s,t,run[N_MAX];

     if (a[1] != 0  || a[n] != 0) return 0;
     r = RLE(a,run,n);

     // Compute j of prefix 21^j
     if (run[1] != 2) return 0;
     j = 0;
     while (run[j+2] == 1 && j+2 <= r) j++;

     // Compute s of prefix (21^j)^s
     s = 1;
     while (s <= r/(1+j) -1 && run[s*(j+1)+1] == 2) {
          for (i=1; i<=j; i++) if (run[s*(j+1)+1+i] != 1) return 0;
          s++;
     }

     // Test remainder of string is (21^j)^s is 1^t
     for (i=s*(j+1)+1; i<=r; i++) if (run[i] != 1) return 0;
     t = r - s*(1+j);

     if (s >= 2 && t >= 2 && j%2 == 0) return 1;
     return 0;
}
// ================================================================================
// Apply PRR^{t+1} to a[1..n] to get b[1..n], where t is the length of the
// prefix before the first 00 or 11 in a[2..n] up to n-2
// ================================================================================
int Shift(int a[], int b[]) {
     int i,t = 0;
     while (a[t+2] != a[t+3] && t < n-2) t++;
     for (i=1; i<=n; i++) b[i] = a[i];
     for (i=1; i<=n; i++) b[i+n] = (b[i] + b[i+1] + b[n+i-1]) % 2;
     for (i=1; i<=n; i++) b[i] = b[i+t+1];
     return t;
}
// ================================================================================
// Test if b[1..len] is the lex largest rep (under rotation), if so, return the
// period p; otherwise return 0. Eg. (411411, p=3)(44211, p=5) (411412, p=0).
// ================================================================================
int IsLargest(int b[], int len) {
     int i, p=1;
     for (i=2; i<=len; i++) {
          if (b[i-p] < b[i]) return 0;
          if (b[i-p] > b[i]) p = i;
     }
     if (len % p != 0) return 0;
     return p;
}
```

```
// ============================================================================
// Membership testers not including the cycle containing 0101010...
// ============================================================================
int RLrep(int a[]) {
    int p,r,rle[N_MAX];

    r = RLE(a,rle,n-1);
    p = IsLargest(rle,r);

    // PCR-related cycle
    if (a[1] == a[n]) {
        if (r == n-1 && a[1] == 1) return 0;  // Ignore root a[1..n] = 1010101..
        if (r == 1) return 1;  // Special case: a[1..n] = 000..0 or 111..1
        if (p > 0 && a[1] != a[n-1] && (p == r || a[1] == 1 || p%2 == 0)) return 1;
    }
    // CCR-related cycle
    if (a[1] != a[n]) {
        if (p > 0 && a[1] == 1 && (a[n-1] == 1)) return 1;
    }
    return 0;
}
// ============================================================================
int LCrep(int a[]) {
    int b[N_MAX];

    if (a[1] != a[2]) return 0;
    Shift(a,b);
    return RLrep(b);
}
// ============================================================================
int SameRep(int a[]) {
    int b[N_MAX];

    Shift(a,b);
    if (Special(a) || (LCrep(a) && !Special(b))) return 1;
    return 0;
}
// ============================================================================
// Repeatedly apply the Prefer-Same or LC or RL successor rule starting with 1^n
// ============================================================================
void DB(int type) {
    int i,j,v,a[N_MAX],REP;

    for (i=1; i<=n; i++) a[i] = 1;  // Initial string

    for (j=1; j<=pow(2,n); j++) {
        printf("%d", a[1]);

        v = (a[1] + a[2] + a[n]) % 2;
        REP = 0;
        // Membership testing of a[1..n]
        if (type == 1 && SameRep(a)) REP = 1;
        if (type == 2 && LCrep(a)) REP = 1;
        if (type == 3 && RLrep(a)) REP = 1;


        // Membership testing of conjugate of a[1..n]
        a[1] = 1 - a[1];
        if (type == 1 && SameRep(a)) REP = 1;
        if (type == 2 && LCrep(a)) REP = 1;
        if (type == 3 && RLrep(a)) REP = 1;

        // Shift String and add next bit
        for (i=1; i<n; i++) a[i] = a[i+1];
        if (REP) a[n] = 1 - v;
        else a[n] = v;
    }
}
//------------------------------------------------------
int main() {
    int type;

    printf("Enter (1) Prefer-same (2) LC (3) RL: ");  scanf("%d", &type);
    printf("Enter n: ");   scanf("%d", &n);

    DB(type);
}
```

B Implementation of the de Bruijn successors $RL2(\omega)$, $LC2(\omega)$, and $O(\omega)$

```
#include<stdio.h>
#include<math.h>
#define N_MAX 50
int n;

// ============================================================================
// Compute the RLE of a[s..m] in run[1..r], returning r = run length
// ============================================================================
int RLE(int a[], int run[], int s, int m) {
     int i,j,r,old;

     old = a[m+1];
     a[m+1] = 1 - a[m];
     r = j = 0;
     for (i=s; i<=m; i++) {
          if (a[i] == a[i+1]) j++;
          else {  run[++r] = j+1;  j = 0;  }
     }
     a[m+1] = old;
     return r;
}
// ============================================================================
// Check if a[1..n] is a "special" RL representative: the RLE of a[1..n] is of
// the form 1 x^j y where y > x and j is odd. Eg. 12224, 1111113 (PCR-related)
// ============================================================================
int Special(int a[]) {
     int i,r,rle[N_MAX];

     r = RLE(a,rle,1,n);
     if (r%2 == 0) return 0;
     for (i=3; i<r; i++) if (rle[i] != rle[2]) return 0;
     if  (a[1] == 1 && a[2] == 0 && i == r && rle[r] > rle[2]) return 1;
     return 0;
}
// ============================================================================
// Apply PRR^{t} to a[1..n] to get b[1..n], where t is the length of the
// prefix in a[1..n] before the first 01 or 10 in a[2..n]
// ============================================================================
int Shift(int a[], int b[]) {
     int i,t=1;

     while (a[t+1] == a[t+2] && t < n-1) t++;
     for (i=1; i<=n; i++) b[i] = a[i];
     for (i=1; i<=n; i++) b[i+n] = (b[i] + b[i+1] + b[n+i-1]) % 2;
     for (i=1; i<=n; i++) b[i] = b[i+t];
     return t;
}
// ============================================================================
// Test if b[1..len] is the lex smallest rep (under rotation), if so, return the
// period p; otherwise return 0. Eg. (114114, p=3)(11244, p=5)(124114, p=0).
// ============================================================================
int IsSmallest(int b[], int len) {
     int i, p=1;
     for (i=2; i<=len; i++) {
          if (b[i-p] > b[i]) return 0;
          if (b[i-p] < b[i]) p = i;
     }
     if (len % p != 0) return 0;
     return p;
}
// ============================================================================
// Membership testers with case for 111111...1  (run length for a[2..n])
// ============================================================================
int RL2rep(int a[]) {
     int p,r,rle[N_MAX];

     r = RLE(a,rle,2,n);
     if (r == 1) return 1;        // Special case: a[1..n] = 000..0 or 111..1
     if (a[1] == a[2]) return 0;
     p = IsSmallest(rle,r);

     if (a[1] == a[n] && p > 0 && (p == r || a[1] == 0 || p%2 == 0)) return 1;  //PCR-related
     if (a[1] != a[n] && p > 0 && a[1] == 0) return 1;  // CCR-related
     return 0;
}
```

```
// ================================================================================
int LC2rep(int a[]) {
    int t,b[N_MAX];

    if (a[1] == a[2]) return 0;
    t = Shift(a,b);
    return RL2rep(b);
}
// ================================================================================
int OppRep(int a[]) {
    int b[N_MAX];

    Shift(a,b);
    if (Special(a) || (LC2rep(a) && !Special(b))) return 1;
    return 0;
}
// ================================================================================
// Repeatedly apply the Prefer Opp or LC or RL successor rule starting with 1^n
// ================================================================================
void DB(int type) {
    int i,j,v,a[N_MAX],REP;

    // Initial string
    for (i=1; i<=n; i+=2) a[i] = 0;
    for (i=2; i<=n; i+=2) a[i] = 1;

    for (j=1; j<=pow(2,n); j++) {
        printf("%d", a[1]);

        v = (a[1] + a[2] + a[n]) % 2;
        REP = 0;
        // Membership testing of a[1..n]
        if (type == 1 && OppRep(a)) REP = 1;
        if (type == 2 && LC2rep(a)) REP = 1;
        if (type == 3 && RL2rep(a)) REP = 1;

        // Membership testing of conjugate of a[1..n]
        a[1] = 1 - a[1];
        if (type == 1 && OppRep(a)) REP = 1;
        if (type == 2 && LC2rep(a)) REP = 1;
        if (type == 3 && RL2rep(a)) REP = 1;

        // Shift String and add next bit
        for (i=1; i<n; i++) a[i] = a[i+1];
        if (REP) a[n] = 1 - v;
        else a[n] = v;
    }
}
//--------------------------------------------------------
int main() {
    int type;

    printf("Enter (1) Prefer-opposite (2) LC2 (3) RL2: ");  scanf("%d", &type);
    printf("Enter n: ");   scanf("%d", &n);

    DB(type);
}
```