A successor rule framework for constructing k-ary de Bruijn sequences and universal cycles

D. Gabric J. Sawada A. Williams D. Wong

July 24, 2018

Abstract

We present a simple framework for constructing k-ary de Bruijn sequences, and more generally, universal cycles, via successor rules. The framework is based on the often used method of joining disjoint cycles. It generalizes several previously known de Bruijn sequence constructions based on the pure cycling register and is applied to derive a new construction that is perhaps the simplest of all successors. Furthermore, it generalizes an algorithm to construct binary de Bruijn sequences based on any arbitrary nonsingular feedback function. The framework is applied to derive and prove the correctness of successors to efficiently construct (i) universal cycles for k-ary strings of length n whose weight is bounded by some w and (ii) universal cycles for permutations. It has also been subsequently been applied to find the first universal cycle constructions for weak orders.

1 Introduction

Let Σ_k denote the alphabet $\{1, 2, \ldots, k\}$ and let Σ_k^n denote the set of all k-ary strings of length n. A *de Bruijn sequence* is a k-ary string of length k^n that when considered cyclicly, contains every string in Σ_k^n as a substring. More generally, given a subset $\mathbf{S}$ of Σ_k^n, a *universal cycle* for $\mathbf{S}$ is a k-ary string of length $|\mathbf{S}|$ that when considered cyclicly contains every string in $\mathbf{S}$ as a substring. In this paper, we present a framework for deriving and proving the correctness of de Bruijn sequence constructions, and more generally, universal cycle constructions, by applying the standard approach of joining disjoint cycles. Each application of the framework yields a *successor rule* which is a function that determines the next symbol in a universal cycle using the previous n symbols.

De Bruijn sequences are well known to be in one-to-one correspondence with Euler cycles in the de Bruijn graph. A downside to applying standard Euler cycle algorithms to construct de Bruijn sequences, such as the ones by Hierholzer [11] and Fleury [7], is that they require $O(k^n)$ memory to store the graph. This has led to a significant amount of disjoint literature for producing a wide variety of space-efficient algorithms to construct de Bruijn sequences. Due to the correspondence with the aforementioned Euler cycles, every construction method will have a corresponding cycle-joining interpretation in the de Bruijn graph (following Hierholzer's approach), even if its correspondence is unknown or hard to determine. The framework presented in this paper generalizes several previously known k-ary de Bruijn sequence constructions including:

1. A concatenation scheme with an implicitly described successor rule by Fredricksen and Maiorana [8],
2. A successor rule based approach by Etzion [5] that can be used to construct an exponential number of de Bruijn sequences.
3. A cycle-joining approach by Yang and Dai [25] that can be applied to any nonsingular feedback function to construct de Bruijn sequences.

4. A very simple successor rule by Wong et al. [21, 22].

5. A concatenation scheme with an explicitly described successor rule by Dragon et al. [4],

Compared to the binary case [6, 16, 17], there are relatively few efficient constructions for an alphabet of arbitrary size. Ralston [18] describes a recursive approach that is based on the aforementioned algorithm by Fredricksen and Maiorana [8]. There are preference based greedy constructions as detailed by Alhakim [1] and a look-up table approach by Xie [24]; however, like the Euler cycle approaches, they require exponential space.

In Section 2 we present our generic successor rule framework for universal cycles and de Bruijn sequences that is very similar in spirit to the approach by Yang and Dai [25]. Then in Section 3 we apply the framework to obtain eight universal cycle successors for sets of strings with a weight constraint. When the weight constraints are removed, the successors construct de Bruijn sequences. In Section 4 we apply the framework to derive four de Bruijn sequence constructions based on any arbitrary nonsingular feedback function. In Section 5 we outline how the framework can be applied to easily produce a shorthand universal cycle for permutations. The framework has also been applied to find the first universal cycle constructions for weak orders [23]. We conclude in Section 6 with some implementation considerations. Our k-ary framework generalizes a simpler *spanning-tree*-like framework for the binary case [9].

2 A successor rule framework

For the remainder of this paper assume $n, k \geq 2$ and that $\mathbf{S}$ is a non-empty subset of Σ_k^n. All arithmetic is considered to be modulo k, where $0 \equiv k$.

Definition 2.1 *A function $f : \Sigma_k^n \to \Sigma_k$ is said to be a* feedback function.

Definition 2.2 *A feedback function f is a* UC-successor *of $\mathbf{S}$ if there exists a universal cycle U for $\mathbf{S}$ such that each string $\alpha \in \mathbf{S}$ is followed by $f(\alpha)$ in U.*

In this definition the domain of f is defined to be Σ_k^n, not $\mathbf{S}$, to simplify some of our upcoming proofs. In the special case where $\mathbf{S} = \Sigma_k^n$ we say a UC-successor is a *de Bruijn-successor*.

Definition 2.3 *A partition of $\mathbf{S}$ into subsets $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ is a* UC-partition with respect to f *if f is a UC-successor for each $\mathbf{S}_i$ where $i \in \{1, 2, \ldots, m\}$.*

Definition 2.4 *Let $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ be an ordered partition of $\mathbf{S}$. For $2 \leq i \leq m$, let $x_i, y_i, z_i \in \Sigma_k$ and let $\beta_i \in \Sigma_k^{n-1}$. A sequence of tuples $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ is a* spanning sequence *of the partition if for each (β_i, x_i, y_i, z_i):*

(i) $y_i\beta_i \in \mathbf{S}_i$,

(ii) *if $i = first(\beta_i)$ then $x_i\beta_i \in \mathbf{S}_j$ for some $j < i$,*

(iii) *$x_iy_iz_i$ is a substring of the cyclic string created by starting with $x_{first(\beta_i)}$ then appending each y_j from tuples (β_j, x_j, y_j, z_j) where $\beta_j = \beta_i$ in increasing order of index j,*

where $first(\beta_i)$ is the smallest index of a tuple containing β_i.

We note the following two remarks with respect to the above definition.

Remark 2.5 *If β_i is distinct amongst all tuples then $x_i = z_i$.*

Remark 2.6 *If $\beta_i = \beta_j$ such that $i < j$ and there is no $i < t < j$ such that $\beta_t = \beta_j$ then $y_i = x_j$ and $z_i = y_j$.*

Example 1 Consider $\mathbf{S} = \Sigma_3^4$ and the feedback function $f(a_1a_2a_3a_4) = a_1+1$. The following partition of $\mathbf{S}$ into sets $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_8$ is a UC-partition with respect to f.

$\mathbf{S}_1$	$\mathbf{S}_2$	$\mathbf{S}_3$	$\mathbf{S}_4$	$\mathbf{S}_5$	$\mathbf{S}_6$	$\mathbf{S}_7$	$\mathbf{S}_8$
1111	1113	1121	1123	1131	1133	1213	1231
1112	1132	1212	1232	1312	1332	2132	2312
1122	1322	2122	2322	3122	3322	1323	3123
1222	3222	1223	3223	1221	3221	3232	
2222	2221	2232	2231	2212	2211	2321	
2223	2213	2323	2313	2123	2113	3213	
2233	2133	3233	3133	1233		2131	
2333	1333	2331	1331	2332		1313	
3333	3332	3313	3312	3323		3132	
3331	3321	3131	3121	3231		1321	
3311	3211	1311	1211	2311		3212	
3111	2111	3112	2112	3113		2121	

The corresponding universal cycles for each part can be obtained by concatenating together the first symbol from each string (top to bottom). Specifically the 8 universal cycle are:

α_1 = 111122223333
α_2 = 111322213332
α_3 = 112122323313
α_4 = 112322313312
α_5 = 113122123323
α_6 = 113322
α_7 = 121323213132
α_8 = 123

The following is a spanning sequence for this partition:

$(\beta_2{=}111, 3, 2, 3)$
$(\beta_3{=}112, 1, 3, 2)$
$(\beta_4{=}112, 3, 2, 1)$
$(\beta_5{=}113, 1, 3, 2)$
$(\beta_6{=}113, 3, 2, 1)$
$(\beta_7{=}121, 3, 2, 3)$
$(\beta_8{=}123, 1, 3, 1)$

In this case, observe that each $y_i\beta_i$ is last string in $\mathbf{S}_i$.

The following lemma describes when two universal cycles can be joined together to create a universal cycle for a larger set. The result follows from [5, Theorem 1] and more directly from [20, Lemma 3].

Lemma 2.7 (Cycle Joining Lemma) *Let $\mathbf{S}_1, \mathbf{S}_2$ be a UC-partition of $\mathbf{S}$ with respect to a feedback function f where $x\beta \in \mathbf{S}_1$ and $y\beta \in \mathbf{S}_2$. Then the following feedback function f' is a UC-successor for $\mathbf{S}$:*

$$f'(\alpha) = \begin{cases} f(x\beta) & \text{if } \alpha = y\beta; \\ f(y\beta) & \text{if } \alpha = x\beta; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

Theorem 2.8 *Let $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ be a UC-partition of $\mathbf{S}$ with respect to f with spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for some $m \geq 2$. Then the following feedback function g is a UC-successor for $\mathbf{S}$:*

$$g(\alpha) = \begin{cases} f(y_i\beta_i) & \text{if } \alpha = x_i\beta_i \text{ for some } i \in \{2, 3, \ldots, m\} \text{ and } i = first(\beta_i); \\ f(z_i\beta_i) & \text{if } \alpha = y_i\beta_i \text{ for some } i \in \{2, 3, \ldots, m\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

Proof. The proof is by induction on m. In the base case when $m = 2$, the spanning sequence is (β_2, x_2, y_2, z_2). By definition, we have $x_2\beta_2 \in \mathbf{S}_1$ and $y_2\beta_2 \in \mathbf{S}_2$, and by Remark 2.5 $x_2 = z_2$. The function $g(\alpha)$ is obtained by applying the Cycle Joining Lemma where $\beta = \beta_2$, $x = x_2$ and $y = y_2$. If $m > 2$, there are two cases depending on whether or not β_m is distinct amongst all tuples in the spanning sequence.

- (β_m is distinct) Since β_m is distinct, $(\beta_2, x_2, y_2, z_2), \ldots, (\beta_{m-1}, x_{m-1}, y_{m-1}, z_{m-1})$ is a spanning sequence for the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{m-1}$ of $\mathbf{S} \setminus \mathbf{S}_m$ with respect to f. Let g' be the UC-successor for $\mathbf{S} \setminus \mathbf{S}_m$ obtained by applying induction. By its definition, g' is equivalent to f for strings in $\mathbf{S}_m$ and thus g' is also a UC-successor for $\mathbf{S}_m$. Since β_m is distinct, by Remark 2.5 $x_m = z_m$. By applying the Cycle Joining Lemma on $\mathbf{S} \setminus \mathbf{S}_m$ and $\mathbf{S}_m$ where $\beta = \beta_m$, $x = x_m$ and $y = y_m$, the resulting UC-successor for $\mathbf{S}$ is equivalent to g.

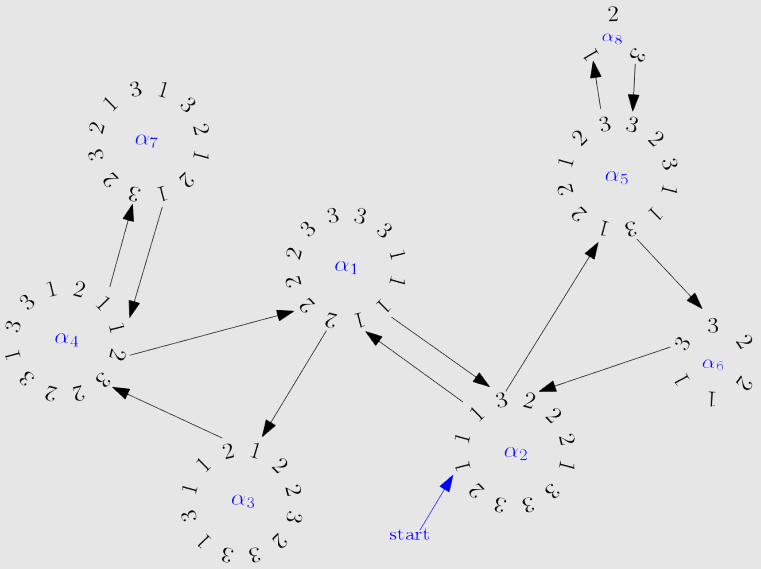
- (β_m is not distinct) Let j be the largest index less than m such that $\beta_j = \beta_m$. From Remark 2.6 $y_j = x_m$ and $z_j = y_m$. Also if i is the smallest index such that $\beta_i = \beta_j = \beta_m$, then $x_i = z_m$ by point (iii) in the definition of a spanning sequence. Thus

  $$(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_j, x_j, y_j, z_m), \ldots, (\beta_{m-1}, x_{m-1}, y_{m-1}, z_{m-1})$$

  is a spanning sequence of the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{m-1}$ of $\mathbf{S} \setminus \mathbf{S}_m$ with respect to f. Let g' be the UC-successor for $\mathbf{S} \setminus \mathbf{S}_m$ obtained by applying induction. By its definition, g' is equivalent to f for strings in $\mathbf{S}_m$ and thus g' is also a UC-successor for $\mathbf{S}_m$. By applying the Cycle Joining Lemma on $\mathbf{S}_1$ and $\mathbf{S}_2$ where $\beta = \beta_m$, and $x = x_m$ and $y = y_m$, the resulting UC-successor for $\mathbf{S}$ is equivalent to g. Note in particular that $g'(y_j\beta_j) = f(z_m\beta_m)$ and that $g(y_j\beta_j) = f(y_m\beta_m) = f(z_j\beta_j)$ and $g(y_m\beta_m) = g'(y_j\beta_j) = f(z_m\beta_m)$.

□

Example 2 Recall the UC-partition of $\mathbf{S} = \Sigma_3^4$ with respect to $f(a_1a_2a_3a_4) = a_1+1$ and its corresponding spanning sequence (from Example 1). The following illustrates the universal cycle construction for $\mathbf{S}$ by applying Theorem 2.8 (or more specifically, the upcoming Theorem 4.3).



The corresponding universal cycle for $\mathbf{S}$ is:

111121223233131123223133121323213132121122223333111312212312332311332211322213332.

A similar UC-successor can be derived by essentially reversing the direction of each β-cycle (see the figure in Example 1). A proof of the following theorem will be identical to the previous proof, except for updating the observations in the final sentence.

Theorem 2.9 *Let $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ be a UC-partition of $\mathbf{S}$ with respect to f with spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for some $m \geq 2$. Then the following feedback function g' is a UC-successor for $\mathbf{S}$:*

$$g'(\alpha) = \begin{cases} f(x_i\beta_i) & \text{if } \alpha = y_i\beta_i \text{ for some } i \in \{2, 3, \ldots, m\} \text{ and } i = first(\beta_i); \\ f(y_i\beta_i) & \text{if } \alpha = z_i\beta_i \text{ for some } i \in \{2, 3, \ldots, m\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

2.1 Simplification for special cases

In this section we provide a simplified definition of a spanning sequence for the special case when each β_i is distinct. This leads to a more restricted, but simplified successor rule result. This result can be further simplified for the binary case when $k = 2$, as described in [9].

Definition 2.10 *Let $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ be an ordered partition of $\mathbf{S}$. For $2 \leq i \leq m$, let $x_i, y_i \in \Sigma_k$ and let $\beta_i \in \Sigma_k^{n-1}$. A sequence of tuples $(\beta_2, x_2, y_2), (\beta_3, x_3, y_3), \ldots, (\beta_m, x_m, y_m)$ is a* simplified spanning sequence *of the partition if each β_i is unique and for each i the string $y_i\beta_i \in \mathbf{S}_i$ and the string $x_i\beta_i \in \mathbf{S}_j$ for some $j < i$.*

The following corollary is a direct consequence of Theorem 2.8 and Remark 2.5.

Corollary 2.11 *Let $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ be a UC-partition of $\mathbf{S}$ with respect to f with simplified spanning sequence $(\beta_2, x_2, y_2), (\beta_3, x_3, y_3), \ldots, (\beta_m, x_m, y_m)$ for some $m \geq 2$. Then the following feedback function h is a UC-successor for $\mathbf{S}$:*

$$h(\alpha) = \begin{cases} f(x_i\beta_i) & \text{if } \alpha = y_i\beta_i \text{ for some } i \in \{2, 3, \ldots, m\}; \\ f(y_i\beta_i) & \text{if } \alpha = x_i\beta_i \text{ for some } i \in \{2, 3, \ldots, m\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

3 UC-successors based on the PCR

Each of the eight successors presented in this section are based on the Pure Cycling Register (PCR) which is defined by the feedback function $f(a_1a_2\cdots a_n) = a_1$. It is well known that the UC-partition of Σ_k^n with respect to the PCR corresponds to equivalence classes of strings under rotation. We call the lexicographically smallest string in each such class a *necklace*. Let $\mathbf{N}_k(n)$ denote the set of all necklaces in Σ_k^n. Let $\mathbf{Neck}(\alpha)$ be the set of strings rotationally equivalent to α. Then $\{\mathbf{Neck}(\alpha) \mid \alpha \in \mathbf{N}_k(n)\}$ is a UC-partition of Σ_k^n with respect to the PCR.

The *weight* of a string α, denoted by $\omega(\alpha)$, is the sum of its symbols. In this section we apply the framework from the previous section to develop four UC-successors for subsets of Σ_k^n whose strings have weight *at most* w. Then we present four UC-successors for subsets of Σ_k^n whose strings have weight *at least* w. When the weight constraints are removed, the UC-successors correspond to de Bruijn-successors which are discussed in more detail at the end of this section.

3.1 Lower bound on weight

First symbol
Let $\alpha = a_1a_2\cdots a_n$ and let $n \leq w \leq kn$. Assume $\omega(\alpha) \geq w$. Let x be the largest symbol in $\Sigma_k \setminus \{k\}$ such that $xa_2a_3\cdots a_n$ is in $\mathbf{N}_k(n)$ and $\omega(xa_2a_3\cdots a_n) \geq w$, or let $x = 0$ if no such symbol exists. Let v be the smallest value in Σ_k such that $\omega(va_2a_3\cdots a_n) \geq w$. Define two functions from Σ_k^n to Σ_k as follows:

$$g_1(\alpha) = \begin{cases} x{+}1 & \text{if } x > 0 \text{ and } a_1 = v; \\ a_1{-}1 & \text{if } x > 0 \text{ and } v < a_1 \leq x{+}1; \\ a_1 & \text{otherwise,} \end{cases}$$

and

$$g_1'(\alpha) = \begin{cases} v & \text{if } x > 0 \text{ and } a_1 = x{+}1; \\ a_1{+}1 & \text{if } x > 0 \text{ and } a_1 < x{+}1; \\ a_1 & \text{otherwise.} \end{cases}$$

Theorem 3.1 *The functions g_1 and g_1' are UC-successors for the subset $\mathbf{S}$ of Σ_k^n consisting of strings whose weight is greater than or equal to some fixed w where $n \leq w \leq kn$.*

Proof. If $w = kn$ then $\mathbf{S} = \{k^n\}$ and $g_1(\alpha) = g_1'(\alpha) = a_1$ which are UC-successors for $\mathbf{S}$. Thus, assume $w < kn$. Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of $\mathbf{S}$ with respect to the PCR. This is well-defined since each subset contains strings with equal weight. Let the subsets be ordered in reverse lexicographic order with respect to their necklace representatives. We construct a spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for the partition. Since the largest necklace representative is k^n and $w < kn$, we have $\mathbf{S}_1 = \{k^n\}$, $m \geq 2$, and for $i \in \{2, 3, \ldots, m\}$ the necklace representative γ_i for $\mathbf{S}_i$ must start with a symbol y_i that is less than k. Let $\beta_i = b_1b_2\cdots b_{n-1}$ and let $\gamma_i = y_i\beta_i$. Since $y_i\beta_i$ is in $\mathbf{S}_i$, condition (i) is satisfied in the definition of a spanning sequence. Let $x_i = y_i + 1$. Observe that the necklace representative of $\mathbf{Neck}(x_i\beta_i)$ is clearly larger than γ_i and has more weight than γ_i. This implies that $x_i\beta_i$ in in some $\mathbf{S}_j$ where $j < i$. Thus (ii) is satisfied in the definition of a spanning sequence. Let x be the largest symbol of $\Sigma_k \setminus \{k\}$ such that $x\beta_i$ is a necklace and $\omega(x\beta_i) \geq w$. Let v be smallest value in Σ_k such that $\omega(v\beta_i) \geq w$. If $y_i > v$, then let $z_i = y_i - 1$; otherwise $y_i = v$ and let $z_i = x{+}1$. Because of the ordering imposed on the UC-partition, condition (iii) is satisfied and we have just constructed a valid spanning sequence for the partition. The UC-successor for $\mathbf{S}$ obtained by applying Theorem 2.8 (respectively, Theorem 2.9) to this UC-partition and spanning sequence is equivalent to g_1 (respectively, g_1'). □

Last non-k (lex least)
Let $\alpha = a_1a_2\cdots a_n$ and let $n \leq w \leq kn$. Assume $\omega(\alpha) \geq w$. If $\alpha = k^n$, let $j = n$; otherwise let j be the smallest index of $a_2a_3\cdots a_n$ such that $a_j \neq k$. Let x be the smallest symbol in Σ_k such that $a_ja_{j+1}\cdots a_nxk^{j-2}$ is in $\mathbf{N}_k(n)$ and $\omega(xa_2a_3\cdots a_n) \geq w$, or let $x = 0$ if no such symbol exists. Define two functions from Σ_k^n to Σ_k as follows:

$$g_2(\alpha) = \begin{cases} k & \text{if } x \neq 0 \text{ and } a_1 = x; \\ a_1{-}1 & \text{if } x \neq 0 \text{ and } a_1 > x; \\ a_1 & \text{otherwise,} \end{cases}$$

and

$$g_2'(\alpha) = \begin{cases} x & \text{if } x \neq 0 \text{ and } a_1 = k; \\ a_1{+}1 & \text{if } x \neq 0 \text{ and } a_1 < k; \\ a_1 & \text{otherwise.} \end{cases}$$

Theorem 3.2 *The functions g_2 and g_2' are UC-successors for the subset $\mathbf{S}$ of Σ_k^n consisting of strings whose weight is greater than or equal to some fixed w where $n \leq w \leq kn$.*

Proof. If $w = kn$ then $\mathbf{S} = \{k^n\}$ and $g_4(\alpha) = g_4'(\alpha) = a_1$ which are UC-successors for $\mathbf{S}$. Thus, assume $w < kn$. Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of $\mathbf{S}$ with respect to the PCR. This is well-defined since each subset contains strings with equal weight. Let the subsets be ordered in reverse lexicographic order with respect to their necklace representatives. We now construct a spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for the partition. Since the largest necklace representative is k^n and $w < kn$, we have $\mathbf{S}_1 = \{k^n\}$, $m \geq 2$, and for $i \in \{2, 3, \ldots, m\}$ the necklace representative $\gamma_i = c_1 c_2 \cdots c_n$ for $\mathbf{S}_i$ must contain a symbol less than k. Let j be the largest index such that $c_j \neq k$ and let $y_i = c_j$. Let $\beta_i = c_{j+1} \cdots c_n c_1 \cdots c_{j-1}$. Since $y_i \beta_i$ is a rotation of γ_i, it is in $\mathbf{S}_i$. Thus condition (i) is satisfied in the definition of a spanning sequence. Let $x_i = y_i + 1$. Observe that the necklace representative of $\mathbf{Neck}(x_i \beta_i)$ is clearly greater than γ_i and has more weight than γ_i. This implies that $x_i \beta_i$ in in some $\mathbf{S}_j$ where $j < i$. Thus (ii) is satisfied in the definition of a spanning sequence. Let x be the smallest symbol of Σ_k such that $c_1 \cdots c_{j-1} x c_{j+1} \cdots c_n$ is a necklace. If $y_i = x$ then let $z_i = k$; otherwise, let $z_i = y_i - 1$. Because of the ordering imposed on the UC-partition, condition (iii) is satisfied and we have just constructed a valid spanning sequence for the partition. The UC-successor for $\mathbf{S}$ obtained by applying Theorem 2.8 (respectively, Theorem 2.9) to this UC-partition and spanning sequence is equivalent to g_2 (respectively, g_2'). □

3.2 Upper bound on weight

Last symbol
Let $\alpha = a_1 a_2 \cdots a_n$ and let $n \leq w \leq kn$. Assume $\omega(\alpha) \leq w$. Let x be the smallest symbol in $\Sigma_k \setminus \{1\}$ such that $a_2 a_3 \cdots a_n x$ is in $\mathbf{N}_k(n)$ and $\omega(x a_2 a_3 \cdots a_n) \leq w$, or let $x = 0$ if no such symbol exists. Let v be the largest value in Σ_k such that $\omega(v a_2 a_3 \cdots a_n) \leq w$. Define two functions from Σ_k^n to Σ_k as follows:

$$g_3(\alpha) = \begin{cases} x{-}1 & \text{if } x > 0 \text{ and } a_1 = v; \\ a_1{+}1 & \text{if } x > 0 \text{ and } x{-}1 \leq a_1 < v; \\ a_1 & \text{otherwise,} \end{cases}$$

and

$$g_3'(\alpha) = \begin{cases} v & \text{if } x > 0 \text{ and } a_1 = x{-}1; \\ a_1{-}1 & \text{if } x > 0 \text{ and } a_1 > x{-}1; \\ a_1 & \text{otherwise.} \end{cases}$$

Theorem 3.3 *The functions g_3 and g_3' are UC-successors for the subset $\mathbf{S}$ of Σ_k^n consisting of strings whose weight is less than or equal to some fixed w where $n \leq w \leq kn$.*

Proof. If $w = n$ then $\mathbf{S} = \{1^n\}$ and $g_3(\alpha) = g_3'(\alpha) = a_1$ which are UC-successors for $\mathbf{S}$. Thus, assume $w > n$. Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of $\mathbf{S}$ with respect to the PCR. This

is well-defined since each subset contains strings with equal weight. Let the subsets be ordered in lexicographic order with respect to their necklace representatives. We construct a spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for the partition. Since the smallest necklace representative is 1^n and $w > n$, we have $\mathbf{S}_1 = \{1^n\}$, $m \geq 2$, and for $i \in \{2, 3, \ldots, m\}$ the necklace representative γ_i for $\mathbf{S}_i$ must end with a symbol y_i that is greater than 1. Let $\beta_i = b_1 b_2 \cdots b_{n-1}$ and let $\gamma_i = \beta_i y_i$. Since $y_i \beta_i$ is a rotation of γ_i, it is in $\mathbf{S}_i$. Thus condition (i) is satisfied in the definition of a spanning sequence. Let $x_i = y_i - 1$. Observe that the necklace representative of $\mathbf{Neck}(x_i \beta_i)$ is clearly less than γ_i and has less weight than γ_i. This implies that $x_i \beta_i$ in in some $\mathbf{S}_j$ where $j < i$. Thus (ii) is satisfied in the definition of a spanning sequence. Let x be the smallest symbol of Σ_k such that $\beta_i x$ is a necklace and $\omega(x\beta_i) \leq w$. Note $x > 1$. Let v be largest value in Σ_k such that $\omega(v\beta_i) \leq w$. If $y_i < v$, then let $z_i = y_i + 1$; otherwise $y_i = v$ and let $z_i = x-1$. Because of the ordering imposed on the UC-partition, condition (iii) is satisfied and we have just constructed a valid spanning sequence for the partition. The UC-successor for $\mathbf{S}$ obtained by applying Theorem 2.8 (respectively, Theorem 2.9) to this UC-partition and spanning sequence is equivalent to g_3 (respectively, g_3'). □

First non-1 (Grandmama)
Let $\alpha = a_1 a_2 \cdots a_n$ and let $n \leq w \leq kn$. Assume $\omega(\alpha) \leq w$. Let j be the largest index of $a_2 a_3 \cdots a_n$ such that $a_j \neq 1$ or $j = 1$ if no such index exists. Let x be the largest symbol in Σ_k such that $1^{n-j} x a_2 \cdots a_j$ is in $\mathbf{N}_k(n)$ and $\omega(x a_2 a_3 \cdots a_n) \leq w$, or let $x = 0$ if no such symbol exists. Define two functions from Σ_k^n to Σ_k as follows:

$$g_4(\alpha) = \begin{cases} 1 & \text{if } x \neq 0 \text{ and } a_1 = x; \\ a_1{+}1 & \text{if } x \neq 0 \text{ and } a_1 < x; \\ a_1 & \text{otherwise,} \end{cases}$$

and

$$g_4'(\alpha) = \begin{cases} x & \text{if } x \neq 0 \text{ and } a_1 = 1; \\ a_1{-}1 & \text{if } x \neq 0 \text{ and } 1 < a_1 \leq x; \\ a_1 & \text{otherwise.} \end{cases}$$

Theorem 3.4 *The functions g_4 and g_4' are UC-successors for the subset $\mathbf{S}$ of Σ_k^n consisting of strings whose weight is less than or equal to some fixed w where $n \leq w \leq kn$.*

Proof. If $w = n$ then $\mathbf{S} = \{1^n\}$ and $g_4(\alpha) = g_4'(\alpha) = a_1$ which are UC-successors for $\mathbf{S}$. Thus, assume $w > n$. Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of $\mathbf{S}$ with respect to the PCR. This is well-defined since each subset contains strings with equal weight. Let the subsets be ordered in lexicographic order with respect to their necklace representatives. We construct a spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for the partition. Since the smallest necklace representative is 1^n and $w > n$, we have $\mathbf{S}_1 = \{1^n\}$, $m \geq 2$, and for $i \in \{2, 3, \ldots, m\}$ the necklace representative $\gamma_i = c_1 c_2 \cdots c_n$ for $\mathbf{S}_i$ must contain a symbol greater than 1. Let j be the smallest index such that $c_j \neq 1$ and let $y_i = c_j$. Let $\beta_i = c_{j+1} \cdots c_n c_1 \cdots c_{j-1}$. Since $y_i \beta_i$ is a rotation of γ_i, it is in $\mathbf{S}_i$. Thus condition (i) is satisfied in the definition of a spanning sequence. Let $x_i = y_i - 1$. Note that the necklace representative of $\mathbf{Neck}(x_i \beta_i)$ is less than γ_i and has less weight than γ_i. This implies that $x_i \beta_i$ in in some $\mathbf{S}_j$ where $j < i$. Thus (ii) is satisfied in the definition of a spanning sequence. Let x be the largest symbol of Σ_k such that $c_1 \cdots c_{j-1} x c_{j+1} \cdots c_n$ is a necklace. If $y_i = x$ then let $z_i = 1$; otherwise, let $z_i = y_i + 1$. Because of the ordering imposed on the UC-partition, condition (iii) is satisfied and we have just constructed a valid

spanning sequence for the partition. The UC-successor for $\mathbf{S}$ obtained by applying Theorem 2.8 (respectively, Theorem 2.9) to this UC-partition and spanning sequence is equivalent to g_4 (respectively, g'_4). □

3.3 De Bruijn-successors

When removing the weight constraint from the previous eight successor rules, we obtain de Bruijn-successors based on the PCR.

Corollary 3.5 *The functions $g_1, g'_1, g_2, g'_2, g_3, g'_3, g_4, g'_4$ are de Bruijn-successors for Σ_k^n.*

The following table illustrates the de Bruijn sequences for $n = 4$ and $k = 3$ constructed from these eight de Bruijn-successors by starting with 1111.

successor	de Bruijn sequence generated by the given successor for $n = 4$ and $k = 3$
g_1	111122231223312333133332323131323321331133223212311232213121321131113222212122112
g'_1	111122223333133323131323233123321331133223122321231123221312132113111322212122112
g_2	111121113112211231132113312121312221223123212331313221323133213332222322332323333
g'_2	111131132113313221323131332133332222322332323331222122312121312321233112211231112
g_3	111122223333233223232221223122112321233123112121313121113221323132113321333133113
g'_3	111133331332133113231322132113111233123212311223122233332332232322221221121313121 2
g_4	111121211221222213221132123213321113121313112312232223132323113312332233133323333
g'_4	111131312131133133332333123322331123132323122322231112121132133212321122132212222

Three of the successor rules correspond to previously known de Bruijn-successors. In particular:

- The de Bruijn-successor g_2 constructs the lexicographically smallest k-ary de Bruijn sequence. A concatenation scheme for this sequence is given in [8]. The proof of their construction implicitly describes g_2, although it is not explicitly stated. The sequence can also be constructed using a prefer-smallest greedy approach.
- The de Bruijn-successor g_3 corresponds to the successor presented in [22].
- The de Bruijn-successor g_4 constructs the *grandmama* k-ary de Bruijn sequence. A concatenation scheme for this sequence is given in [3] which also includes a successor rule that is equivalent to g_4.

By removing the weight constraints, the statements of the de Bruijn-successors can be simplified. In particular, we re-state the new de Bruijn-successor g_1, which is also the simplest of all the successors.

First symbol (no weight constraint)
Let $\alpha = a_1 a_2 \cdots a_n$. Let x be the largest symbol in $\Sigma_k \setminus \{k\}$ such that $x a_2 a_3 \cdots a_n$ is in $\mathbf{N}_k(n)$, or let $x = 0$ if no such symbol exists. Then g_1 can be restated as:

$$g_1(\alpha) = \begin{cases} x{+}1 & \text{if } x > 0 \text{ and } a_1 = 1; \\ a_1{-}1 & \text{if } x > 0 \text{ and } 1 < a_1 \leq x{+}1; \\ a_1 & \text{otherwise.} \end{cases}$$

4 De Bruijn-successors for an arbitrary feedback function

In this section we generalize two results from the previous section based on the PCR to an entire class of feedback functions.

Definition 4.1 *A feedback function f is said to be* nonsingular *if the function $F : \Sigma_k^n \to \Sigma_k^n$ defined as $F(a_1a_2\cdots a_n) = a_2a_3\cdots a_nf(a_1a_2\cdots a_n)$ is one-to-one.*

Necessary and sufficient conditions for when k-ary feedback functions are nonsingular are given by Lai [15]. In the binary case, a feedback function is nonsingular if and only if it is of the form $f(a_1a_2\cdots a_n) = a_1 + f_0(a_2a_3\cdots a_n)$ where f_0 is any function that maps length $n-1$ binary strings to $\{0,1\}$ [10].

First, we generalize the *Last symbol* approach which in turn generalizes the binary de Bruijn sequence constructions give by Jansen, Franx, and Boekee [13]. Then we generalize the *First non-1* approach. In each case let the representative of each part (cycle) induced by the nonsingular feedback function f be its lexicographically smallest string and let $\mathbf{Reps}(f)$ denote the set containing each of these representatives.

4.1 Last symbol

In this section we apply our successor rule framework by focussing on the last symbol of each string in $\mathbf{Reps}(f)$.

Definition 4.2 *Let $\beta \in \Sigma_k^{n-1}$. Define $\tau(\beta)$ to be the increasing sequence of symbols $x \in \Sigma_k$ such that $\beta x \in \mathbf{Reps}(f)$ with one possible addition: (a) if 1 is already in the sequence and $\beta 1 \neq 1^n$, then prepend $f(1\beta)$ to the front or (b) if 1 is not in this sequence and the sequence is non-empty, then prepend 1 to the front. In the special case when $\beta = 1^{n-1}$ and $x = 1$ is the only symbol in Σ_k such that $\beta x \in \mathbf{Reps}(f)$, define $\tau(\beta)$ to be empty.*

Note that if $\beta 1 \neq 1^n$ and $v = f(1\beta)$ then $1\beta < \beta v$, and hence $\beta v \notin \mathbf{Reps}(f)$. Thus, each symbol in $\tau(\beta)$ is unique. Also note that by this definition $\tau(\beta)$ will never have only one symbol.

Example 3 Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_8$ of Σ_3^4 with respect to $f(a_1a_2a_3a_4) = a_1+1$ ordered lexicographically based on the lexicographically smallest string as representative:

$\mathbf{S}_1$	$\mathbf{S}_2$	$\mathbf{S}_3$	$\mathbf{S}_4$	$\mathbf{S}_5$	$\mathbf{S}_6$	$\mathbf{S}_7$	$\mathbf{S}_8$
1111	**1113**	**1121**	**1123**	**1131**	**1133**	**1213**	**1231**
1112	1132	1212	1232	1312	1332	2132	2312
1122	1322	2122	2322	3122	3322	1323	3123
1222	3222	1223	3223	1221	3221	3232	
2222	2221	2232	2231	2212	2211	2321	
2223	2213	2323	2313	2123	2113	3213	
2233	2133	3233	3133	1233		2131	
2333	1333	2331	1331	2332		1313	
3333	3332	3313	3312	3323		3132	
3331	3321	3131	3121	3231		1321	
3311	3211	1311	1211	2311		3212	
3111	2111	3112	2112	3113		2121	

Since $\mathbf{Reps}(f) = \{1111, 1113, 1121, 1123, 1131, 1133, 1213, 1231\}$, based on the definition of $\tau(\beta)$ we have

$$\tau(111) = \langle 1,3\rangle,\ \tau(112) = \langle 2,1,3\rangle,\ \tau(113) = \langle 2,1,3\rangle,\ \tau(121) = \langle 1,3\rangle,\ \tau(123) = \langle 2,1\rangle,$$

and for all other β, $\tau(\beta)$ is empty.

The set of sequences τ effectively describe how universal cycles for the UC-partition of Σ_k^n with respect to f can be joined together using our successor rule framework.

Generalized last symbol
Let $\alpha = a_1a_2\cdots a_n$. Let $\tau(a_2a_3\cdots a_n) = t_1, t_2, \ldots, t_p$ be considered cyclicly. Define $g_5 : \Sigma_k^n \to \Sigma_k$ as follows:

$$g_5(\alpha) = \begin{cases} t_{j+1} & \text{if } f(\alpha) = t_j \text{ for some } j \in \{1, 2, \ldots, p\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

Define $g_5' : \Sigma_k^n \to \Sigma_k$ as follows:

$$g_5'(\alpha) = \begin{cases} t_{j-1} & \text{if } f(\alpha) = t_j \text{ for some } j \in \{1, 2, \ldots, p\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

Theorem 4.3 *The functions $g_5(\alpha)$ and $g_5'(\alpha)$ are de Bruijn-successors for Σ_k^n.*

Proof. Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of Σ_k^n with respect to a nonsingular feedback function f. Let the subsets be ordered in lexicographic order with respect to their cycle representatives. We construct a spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for the partition. Since 1^n is the smallest string in Σ_k^n, it must be the cycle representative of $\mathbf{S}_1$, and for $i \in \{2, 3, \ldots, m\}$ the cycle representative $\gamma_i = c_1c_2\cdots c_n$ for $\mathbf{S}_i$ must contain an symbol greater than 1. Let $\beta_i = c_1c_2\cdots c_{n-1}$. Let $\tau(\beta_i) = t_1, t_2, \ldots, t_p$ and let $t_j = c_n$. As noted earlier, $p \neq 1$. Define the cyclic string $\sigma = s_1s_2\cdots s_p$ where s_j is the symbol of Σ_k such that $f(s_j\beta) = t_j$. This symbol is well-defined since f is nonsingular. Then let $(x_i, y_i, z_i) = (s_{j-1}, s_j, s_{j+1})$. Since $f(y_i\beta_i) = c_n$, $y_i\beta_i \in \mathbf{S}_i$ and thus condition (i) is satisfied in the definition of a spanning sequence. Since $\gamma_i \neq 1^n$, by the definition for each of the three cases for $\tau'(\beta_i)$, the representative of the subset containing $x_i\beta_i$ will be less than γ_i. Thus (ii) is satisfied in the definition of a spanning sequence. Finally, because of the ordering imposed on the UC-partition and the definition of $\sigma(\beta_i)$, condition (iii) is satisfied. By applying Theorem 2.8 and Theorem 2.9 to this UC-partition and spanning sequence and simplifying the resulting functions we obtain the de Bruijn-successors g_5 and g_5'. □

When this theorem is applied to Σ_3^4 with feedback function $f(a_1a_2a_3a_4) = a_1+1$, the resulting function $g_5(\alpha)$ produces the de Bruijn sequence outlined in Example 2.

4.2 First non-1

In this section we apply our successor rule framework to generalize the binary de Bruijn-successors given in Section 3 based on focussing on the first non-1 of each string in $\mathbf{Reps}(f)$. Let $\alpha = a_1a_2\cdots a_n$. Let $F(\alpha) = a_2\cdots a_nf(\alpha)$ and define $F^j(\alpha)$ recursively to be $F^{j-1}(F(\alpha))$ where $F^1(\alpha) = F(\alpha)$ and $F^0(\alpha) = \alpha$.

Definition 4.4 *Let $\beta = b_1b_2\cdots b_{n-1}$ and let j be the largest integer such that 1^j is a suffix of β. Define $\tau'(\beta)$ to be the increasing sequence of symbols $x \in \Sigma_k \setminus \{1\}$ such that $F^{n-j-1}(\beta x) \in \mathbf{Reps}(f)$ with 1 prepended to the front if the sequence is non-empty.*

Based on this definition observe that $\tau'(\beta)$ will never have only one symbol.

Example 4 Recall the cycle representatives $\mathbf{Reps}(f) = \{1111, 1113, 1121, 1123, 1131, 1133, 1213, 1231\}$ for the UC-partition of Σ_3^4 with respect to the nonsingular feedback function $f(a_1a_2a_3a_4) = a_1+1$ presented in Example 3. Based on the definition of $\tau(\beta)$ we have

$$\tau'(111) = \langle 1,3\rangle,\ \tau'(211) = \langle 1,2,3\rangle,\ \tau'(231) = \langle 1,2\rangle,\ \tau'(311) = \langle 1,2,3\rangle,\ \tau'(321) = \langle 1,2\rangle,$$

and for all other β, $\tau'(\beta)$ is empty. As further example, for $\beta = 231$ note that $j = 1$ is the largest integer such that 1^j is a suffix of β and $F^2(\beta\mathbf{2}) = 1231$ which is in $\mathbf{Reps}(f)$.

The set of sequences τ' effectively describe how universal cycles for the UC-partition of Σ_k^n with respect to f can be joined together using our successor rule framework.

Generalized first non-1
Let $\alpha = a_1a_2\cdots a_n$. Let $\tau'(a_2a_3\cdots a_n) = t_1, t_2, \ldots, t_p$ be considered cyclicly. Define $g_6 : \Sigma_k^n \to \Sigma_k$ as follows:

$$g_6(\alpha) = \begin{cases} t_{j+1} & \text{if } f(\alpha) = t_j \text{ for some } j \in \{1,2,\ldots,p\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

Define $g_6' : \Sigma_k^n \to \Sigma_k$ as follows:

$$g_6'(\alpha) = \begin{cases} t_{j-1} & \text{if } f(\alpha) = t_j \text{ for some } j \in \{1,2,\ldots,p\}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

Theorem 4.5 *The functions $g_6(\alpha)$ and $g_6'(\alpha)$ are de Bruijn-successors for Σ_k^n.*

Proof. Consider the UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of Σ_k^n with respect to a nonsingular feedback function f. Let the subsets be ordered in lexicographic order with respect to their cycle representatives. We construct a spanning sequence $(\beta_2, x_2, y_2, z_2), (\beta_3, x_3, y_3, z_3), \ldots, (\beta_m, x_m, y_m, z_m)$ for the partition. Since 1^n is the smallest string in Σ_k^n, it must be the cycle representative of $\mathbf{S}_1$, and for $i \in \{2,3,\ldots,m\}$ the cycle representative γ_i for $\mathbf{S}_i$ must contain a symbol greater than 1. Let j be the largest integer (it must be less than n) such that 1^j is a prefix of γ_i. Let $\gamma_i' = c_1c_2\cdots c_n$ be the unique string such that $F^{n-j-1}(\gamma_i') = \gamma_i$. Let $\beta_i = c_1c_2\cdots c_{n-1}$. Let $\tau'(\beta_i) = t_1, t_2, \ldots, t_p$ and let $t_j = c_n$. As noted earlier, $p \neq 1$. Define the cyclic string $\sigma = s_1s_2\cdots s_p$ where s_j is the symbol of Σ_k such that $f(s_j\beta) = t_j$. This symbol is well-defined since f is nonsingular. Then let $(x_i, y_i, z_i) = (s_{j-1}, s_j, s_{j+1})$. Since $f(y_i\beta_i) = c_n$, $y_i\beta_i \in \mathbf{S}_i$ and thus condition (i) is satisfied in the definition of a spanning sequence. Since $\gamma_i \neq 1^n$, by the definition for each of the three cases for $\tau'(\beta_i)$, the representative of the subset containing $x_i\beta_i$ will be less than γ_i. Thus (ii) is satisfied in the definition of a spanning sequence. Finally, because of the ordering imposed on the UC-partition and the definition of $\sigma(\beta_i)$, condition (iii) is satisfied. By applying Theorem 2.8 and Theorem 2.9 to this UC-partition and spanning sequence and simplifying the resulting functions we obtain the de Bruijn-successors g_6 and g_6'. □

5 Shorthand universal cycles for permutations

It is easy to demonstrate that universal cycles for permutations in their standard notation do not exist [14]. However, there are several known universal cycle constructions for permutations using a shorthand notation [19, 12]. Let $\pi = p_1p_2\cdots p_n$ be a permutation of order n. A *shorthand permutation* for π is given by

$p_1p_2\cdots p_{n-1}$, where the missing symbol p_n is easily determined. Let $\mathbf{SP}(n)$ denote the set of all shorthand permutations of order n. For example,

$$\begin{aligned}\mathbf{SP}(4) &= 123, 124, 132, 134, 142, 143, 213, 214, 231, 234, 241, 243\\ &\quad 312, 314, 321, 324, 341, 342, 412, 413, 421, 423, 431, 432.\end{aligned}$$

An example of a universal cycle for $\mathbf{SP}(4)$ using our upcoming construction is: 123124132143243142134234. It differs from the previous constructions of [19, 12] in which n appears periodically at every n-th position.

An *inversion* of a permutation π is an ordered pair (p_i, p_j) such that $i < j$ and $p_i > p_j$.

Definition 5.1 *Let $\pi = p_1p_2\cdots p_n$ be a permutation of order n. Consider the rotation $\pi' = q_1q_2\cdots q_n$ of π that starts with the symbol 1. Let j be the smallest index in π' such that there exists an inversion (q_i, q_j) for some $i < j$. Define $inv(\pi) = q_j$.*

As an example, if $\pi = 634215$, then $\pi' = q_1q_2\cdots q_6 = 156342$ and $inv(\pi) = q_4 = 3$. We apply our successor rule framework to $\mathbf{SP}(n)$ using this definition.

Shorthand permutation successor
Let $\pi = p_1p_2\cdots p_{n-1}$ be a shorthand permutation of order n and let z be the missing symbol. Define $g_7 : \mathbf{SP}(n) \to \{1, 2, 3, \ldots, n\}$ as follows:

$$g_7(\pi) = \begin{cases} z & \text{if } (z = n \text{ and } p_1 = inv(\pi)) \text{ or } (p_1 = n \text{ and } z = inv(zp_2p_3\cdots p_{n-1}));\\ z & \text{if } z = p_1 - 1 \text{ or } z = p_1 + 1;\\ p_1 & \text{otherwise.}\end{cases}$$

Theorem 5.2 *The function g_7 is a UC-successor for $\mathbf{SP}(n)$.*

Proof. Since $\mathbf{SP}(n)$ is closed under rotation, consider its UC-partition $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ with respect to the PCR. Let the representatives of each part be the lexicographically smallest string. Order the subsets based on these representatives in increasing order by the number of inversions, then by lexicographic order. Thus, the representative of $\mathbf{S}_1$ is $12\cdots(n-1)$. Every other representative must either contain n or have at least one inversion. We construct a *simplified spanning sequence* $(\beta_2, x_2, y_2), (\beta_3, x_3, y_3), \ldots, (\beta_m, x_m, y_m)$ for the partition. For $i \in \{2, 3, \ldots, m\}$ consider the representative $\pi_i = p_1p_2\cdots p_{n-1}$ for $\mathbf{S}_i$ with missing symbol z. Consider two cases depending on whether or not the missing symbol is n.

- Suppose $z = n$. Then π_i is a permutation of order $n-1$. Let $y_i = inv(\pi_i)$. Define β_i such that $y_i\beta_i$ is a rotation of π_i. Clearly $y_i\beta_i \in \mathbf{S}_i$. Let $x_i = z$. Observe that the representative of $x_i\beta_i$ will be π_i with the symbol y_i replaced with n. Thus, its representative has fewer inversions than π_i which implies that $x_i\beta_i$ belongs to some $\mathbf{S}_j$ where $j < i$.
- Suppose $z \neq n$. Let $y_i = (z+1)$. Define β_i such that $y_i\beta_i$ is a rotation of π_i. Clearly $y_i\beta_i \in \mathbf{S}_i$. Let $x_i = z$. Note that the representative of $x_i\beta_i$ will be π_i with the symbol $(z+1)$ replaced with z. Thus, its representative has the same number of inversions as π_i, but is smaller lexicographically. This implies that $x_i\beta_i$ belongs to some $\mathbf{S}_j$ where $j < i$.

Consider some β_i and its missing symbol z based on the two cases above. If $z = n$, then β_i contains neither n nor $y_i \neq n-1$ based on the definition of $inv(\pi_i)$. Otherwise, β_i contains neither z nor $y_i = z+1$. Thus, if $\beta_i = \beta_j$ for some $2 \leq i < j \leq m$, then they must have the same missing symbol which implies that $y_i = y_j$.

However this implies $y_i\beta_i = y_j\beta_j$ which contradicts $i \neq j$. Thus each β_i is unique, and we have satisfied the conditions in the definition of a simplified spanning sequence.

A UC-successor for $\mathbf{SP}(n)$ is obtained from this simplified spanning sequence by applying Corollary 2.11. When simplified, it corresponds to g_7. The first line of g_7 corresponds to the case outlined above when $z = n$, and the second line corresponds to the case when $z \neq n$. □

6 Implementation considerations

Testing whether or not a string $\alpha = a_1a_2 \cdots a_n$ is a necklace can be tested in $O(n)$ time using $O(n)$ space [2]. By naïvely applying such an algorithm, the values x and v from the eight UC-successors presented in Section 3 can be computed in $O(kn)$ time. It is not difficult to improve the running time of each successor to $O(n)$ time by performing preliminary scans of the relevant string to restrict the possible values for x or v to two choices. We omit the details.

Theorem 6.1 *The UC-successors $g_1, g'_1, g_2, g'_2, g_3, g'_3, g_4$, and g'_4 can be computed in $O(n)$ time using $O(n)$ space.*

Using a similar approach, we can test whether a string α is the lexicographically smallest in an equivalence class $\mathbf{S}$ induced by a nonsingular feedback function. If the feedback function can be computed in $O(1)$ time, then by computing a universal cycle for $\mathbf{S}$ this test can be done in $O(|\mathbf{S}|)$ time using $O(|\mathbf{S}|)$ space. Alternatively, if $|\mathbf{S}|$ is large, then this test can be performed in $O(n|\mathbf{S}|)$ time using $O(n)$ space by applying f to compute successive strings in the equivalence class and comparing them to α.

Theorem 6.2 *Let f be a nonsingular feedback function. If the largest set in the UC-partition of Σ_k^n with respect to f has size C, then the de Bruijn-successors g_5, g'_5, g_6, and g'_6 can construct de Bruijn sequences in $O(knC)$ time using $O(n)$ space. Alternatively, if f can be computed in $O(1)$ time, then the successors can be computed in $O(kC)$ time using $O(C)$ space.*

The shorthand permutation successor g_7 requires that the $O(n)$ time function inv be computed only if the permutation starts with n or its missing symbol is n. Otherwise the function can be computed in $O(1)$ time. By using a circular array to store the current shorthand permutation, and amortizing the work required by the function inv, we obtain the following result.

Theorem 6.3 *The function g_7 can be used to compute a shorthand universal cycle for $\mathbf{SP}(n)$ in $O(1)$-amortized time per symbol using $O(n)$ space.*

References

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

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

[3] 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 Journal of Combinatorics*, 72:1–11, 2018.

[4] P. B. Dragon, O. I. Hernandez, and A. Williams. The grandmama de Bruijn sequence for binary strings. In *Proceedings of LATIN 2016: Theoretical Informatics: 12th Latin American Symposium, Ensenada, Mexico*, pages 347–361. Springer Berlin Heidelberg, 2016.

[5] T. Etzion. An algorithm for constructing m-ary de Bruijn sequences. *Journal of Algorithms*, 7(3):331 – 340, 1986.

[6] T. Etzion and A. Lempel. Algorithms for the generation of full-length shift- register sequences. *IEEE Transactions on Information Theory*, 30(3):480–484, May 1984.

[7] M. Fleury. Deux problèmes de géométrie de situation. *Journal de Mathématiques Élémentaires*, pages 257–261, 1883.

[8] H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Mathematics*, 23(3):207–210, 1978.

[9] D. Gabric, J. Sawada, A. Williams, and D. Wong. A framework for constructing de Bruijn sequences via simple successor rules. *Discrete Mathematics, to appear*, 2018.

[10] S. W. Golomb and L. R. Welch. Nonlinear shift-register sequences. *Jet Prop. Lab., Pasadena, CA, Memo*, pages 20–149, 1957.

[11] C. Hierholzer. Ueber die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren. *Mathematische Annalen*, 6(1):20–32, 1873.

[12] A. E. Holroyd, F. Ruskey, and A. Williams. Shorthand universal cycles for permutations. *Algorithmica*, 64(2):215–245, 2012.

[13] C. J. A. Jansen, W. G. Franx, and D. E. Boekee. An efficient algorithm for the generation of DeBruijn cycles. *IEEE Transactions on Information Theory*, 37(5):1475–1478, Sep 1991.

[14] J. R. Johnson. Universal cycles for permutations. *Discrete Mathematics*, 309(17):5264–5270, 2009.

[15] X. Lai. Condition for the nonsingularity of a feedback shift-register over a general finite field. *IEEE Transactions on Information Theory*, 33(5):747–749, Sep 1987.

[16] C. Li, X. Zeng, C. Li, and T. Helleseth. A class of de Bruijn sequences. *IEEE Trans. Inform. Theory*, 60(12):7955–7969, 2014.

[17] C. Li, X. Zeng, C. Li, T. Helleseth, and M. Li. Construction of de Bruijn sequences from LFSRs with reducible characteristic polynomials. *IEEE Trans. Inform. Theory*, 62(1):610–624, 2016.

[18] A. Ralston. A new memoryless algorithm for de Bruijn sequences. *J. Algorithms*, 2(1):50–62, 1981.

[19] F. Ruskey and A. Williams. An explicit universal cycle for the $(n-1)$-permutations of an n-set. *ACM Transactions on Algorithms (TALG)*, 6(3):45, 2010.

[20] J. Sawada, A. Williams, and D. Wong. Universal cycles for weight-range binary strings. In *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, Revised Selected Papers*, volume 8288 of *Lecture Notes in Computer Science*, pages 388–401. Springer, 2013.

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

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

[23] J. Sawada and D. Wong. Universal cycle constructions for weak orders. *manuscript*, 2018.

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

[25] J.-H. Yang and Z.-D. Dai. Construction of m-ary de Bruijn sequences (extended abstract). In J. Seberry and Y. Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, pages 357–363, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

Appendix - C code for constructing de Bruijn sequences applying g_5, g_5', g_6, or g_6'

```
#include<stdio.h>
#define MAX_N 50

int n,k,a[MAX_N];

// ================================================================================================
int Mod(int x) {

     while (x < 1) x+=k;
     while (x > k) x-=k;
     return x;
}
// ================================================================================================
int f(int a[]) {

     return ( Mod(a[1] + 1) );  // INSERT ANY NONSINGULAR FEEDBACK FUNCTION
}
// ================================================================================================
int Ones(int a[]) {
     for (int i=1; i<=n; i++) if (a[i] != 1) return 0;
     return 1;
}
// ================================================================================================
// Return TRUE iff b[1..n] is a cycle rep = lex smallest string in cycle
// ================================================================================================
int IsRep(int b[]) {
     int i, new_bit, cycle[MAX_N];

     for (i=1; i<=n; i++) cycle[i] = b[i];

     while (1) {
          // Shift and add new bit until returning to b[]
          new_bit = f(cycle);
          for (i=1; i<=n; i++) cycle[i] = cycle[i+1];
          cycle[n] = new_bit;

          // Compare b[] to another in the cycle
          for (i=1; i<=n; i++) {
               if (b[i] < cycle[i]) break;
               if (b[i] > cycle[i]) return 0;
          }
          if (i > n) return 1; // Back to initial string b[]
     }
}
// ================================================================================================
// Compute tau[] and return its size
// ================================================================================================
int TauLastSymbol(int a[], int tau[]) {
     int i,p=0,b[MAX_N];

     // Shift and try all values for b[n]
     for (i=1; i<=n; i++) b[i] = a[i+1];

     for (i=1; i<=k; i++) {
          b[n] = i;
          if (IsRep(b)) {
               if (i == 1 && !Ones(b)) {
                    a[1] = 1;
                    tau[++p] = f(a);   // a[1] is never used again, so no need to restore
               }
               else if (i > 1 && p == 0) tau[++p] = 1;
               tau[++p] = i;
          }
     }
     return p;
}
```

```
// ============================================================================================
int LastSymbol(int a[]) {
    int tau[MAX_N],i,j,v;

    v = f(a);
    j = TauLastSymbol(a,tau);

    for (i=1; i<=j; i++) {
        if (v == tau[i] && i < j)  return tau[i+1];
        if (v == tau[i] && i == j) return tau[1];
    }
    return v;
}
// ============================================================================================
int LastSymbol2(int a[]) {
    int tau[MAX_N],i,j,v;

    v = f(a);
    j = TauLastSymbol(a,tau);

    for (i=1; i<=j; i++) {
        if (v == tau[i] && i == 1)  return tau[j];
        if (v == tau[i] && i > 1)   return tau[i-1];
    }
    return v;
}
// ============================================================================================
// Compute tau'[] and return its size
// ============================================================================================
int TauFirstNonOne(int a[], int tau[]) {
    int i,v,t,j=0,p=0,b[MAX_N];

    for (i=1; i<=n; i++) b[i] = a[i];

    // Shift the j 1s in the suffix to front of string
    while (j < n && b[n-j] == 1) j++;
    for (i=1; i<=n-j; i++) {
        v = f(b);
        for (t=1; t<n; t++) b[t] = b[t+1];
        b[n] = v;
    }
    if (j == n) return 0;

    // Try all values > 1 at position j+1 to see if it is a representative
    p=0;
    for (i=2; i<=k; i++) {
        b[j+1] = i;
        if (IsRep(b)) {
            if (p == 0) tau[++p] = 1;
            tau[++p] = i;
        }
    }
    return p;
}
// ============================================================================================
int FirstNonOne(int a[]) {
    int tau[MAX_N],i,j,v;

    v = f(a);
    j = TauFirstNonOne(a,tau);

    for (i=1; i<=j; i++) {
        if (v == tau[i] && i < j)  return tau[i+1];
        if (v == tau[i] && i == j) return tau[1];
    }
    return v;
}
// ============================================================================================
```

```
int FirstNonOne2(int a[]) {
    int tau[MAX_N],i,j,v;

    v = f(a);
    j = TauFirstNonOne(a,tau);

    for (i=1; i<=j; i++) {
        if (v == tau[i] && i == 1)  return tau[j];
        if (v == tau[i] && i > 1)   return tau[i-1];
    }
    return v;
}
// ================================================================================================
// Generate de Bruijn sequences by iteratively applying a successor rule h() or h2()
// ================================================================================================
void DB(int type) {
    int i, new_bit;

    // Initialize first n bits to 1^n - could start with any string changing the termination
    for (i=0; i<=n; i++)  a[i] = 1;
    do {
        printf("%d", a[1]);
        if (type == 1) new_bit = LastSymbol(a);
        if (type == 2) new_bit = LastSymbol2(a);
        if (type == 3) new_bit = FirstNonOne(a);
        if (type == 4) new_bit = FirstNonOne2(a);

        // Shift and add new bit
        for (i=1; i<=n; i++) a[i] = a[i+1];
        a[n] = new_bit;

    } while (!Ones(a));
}
// ================================================================================================
int main() {

    printf("Enter n: ");    scanf("%d", &n);
    printf("Enter k: ");    scanf("%d", &k);
    printf("Last symbol (incr):\n");  DB(1);  printf("\n\n");
    printf("Last symbol (decr):\n");  DB(2);  printf("\n\n");
    printf("First non-1 (incr):\n");  DB(3);  printf("\n\n");
    printf("First non-1 (decr):\n");  DB(4);  printf("\n\n");
}
```