Universal cycles for k-subsets and k-multisets of an n-set

Colin Campbell
University of Guelph, Canada

Luke Janik-Jones
University of Guelph, Canada

Joe Sawada
University of Guelph, Canada

Abstract

A *universal cycle* for a set $\mathbf{S}$ of combinatorial objects is a cyclic sequence of length $|\mathbf{S}|$ that contains a representation of each element in $\mathbf{S}$ exactly once as a substring. If $\mathbf{S}$ is the set of k-subsets of $[n] = \{1, 2, \ldots, n\}$, it is well-known that universal cycles do not always exists when applying a simple string representation, where 12 or 21 could represent the subset $\{1, 2\}$. Similarly, if $\mathbf{S}$ is the set of k-multisets of $[n]$, it is also known that universal cycles do not always exist using a similar representation, where 112, 121, or 211 could represent the multiset $\{1, 1, 2\}$. In this paper we consider new representations for each of these sets and demonstrate the existence of universal cycles for all $n, k \geq 2$. Furthermore, we provide successor-rule algorithms to construct such universal cycles in $\mathcal{O}(n)$ time per symbol using $\mathcal{O}(n)$ space and demonstrate that necklace concatenation algorithms allow the same sequences to be generated in $\mathcal{O}(1)$ amortized time per symbol. They are the first known universal cycle constructions for k-multisets. The results are obtained by considering constructions for bounded-weight de Bruijn sequences.

1 Introduction

A *universal cycle* for a set $\mathbf{S}$ of combinatorial objects is a cyclic sequence of length $|\mathbf{S}|$ that contains a *representation* of each element in $\mathbf{S}$ exactly once as a substring. In this paper we focus on two sets: (a) $\mathbf{S}_k(n)$, which denotes the set of k-subsets of $[n] = \{1, 2, \ldots, n\}$, and (b) $\mathbf{M}_k(n)$, which denotes the set of k-multisets of $[n]$. For example:

- $\mathbf{S}_2(3) = \Big\{\{1, 2\}, \{1, 3\}, \{2, 3\}\Big\}$, and
- $\mathbf{M}_2(3) = \Big\{\{1, 1\}, \{1, 2\}, \{1, 3\}, \{2, 2\}, \{2, 3\}, \{3, 3\}\Big\}$.

As we demonstrate in this paper, the choice of representation for a k-subset or k-multiset is critical to the existence of universal cycle constructions for these sets.

Subsets. Universal cycles for $\mathbf{S}_k(n)$ have been studied primarily by considering a *standard* string representation, i.e., the subset $\{1, 2\}$ can be represented by either 12 or 21 (see [CDG92, JSH09, Lan12, Rud13]). This representation has a major drawback as universal cycles exist only if n divides $\binom{n}{k}$, or equivalently, if k divides $\binom{n-1}{k-1}$. When this condition is met, Chung, Diaconis, and Graham [CDG92] conjectured that universal cycles for $\mathbf{S}_k(n)$ exist for sufficiently large n once k is fixed. The conjecture was verified for $k = 3, 4, 6$ in [Jac93, Hur94], and universal cycles for subsets were shown not to exist when $k = n - 2$ [SBE+02]. Recently, this conjecture was answered in the affirmative by Glock et al. [GJKO20] by studying Euler tours in hypergraphs. The vast number of cases where universal cycles for $\mathbf{S}_k(n)$ do not exist using this standard representation has led to the study of subset packings and coverings [CHHM09, DL16].

Another common representation for a k-subset is a length-n binary string with exactly k ones. It is straightforward to observe that non-trivial universal cycles do not exist using this representation. However, by considering a *shorthand* binary string representation (one that omits the final redundant symbol), each k-subset corresponds

to a unique length $n-1$ substring with either $k-1$ or k ones. Universal cycles for these strings, and hence k-subsets, can be constructed via a necklace concatenation approach [RSW12] in $\mathcal{O}(1)$ amortized time using $\mathcal{O}(n)$ space [SW13].[1] The same sequence can also be generated by an $\mathcal{O}(n)$ time per symbol successor rule [SW23] based on the "missing symbol register".

In this paper, we consider a novel *difference* representation that represents a k-subset with a length-k string where the first symbol is the smallest element of the subset, and each successive symbol in the string is added to the previous symbol to obtain the next largest symbol in the subset. For example, the subset $\{1,3,4\}$ for $n=5$ can be represented by 121. The *weight* of a string is the sum of its symbols. Observe that by applying the difference representation, the set of k-subsets corresponds to all strings over the alphabet $\{1,2,\ldots,n-k+1\}$ of length k, where the weight of each string is bounded above by n. A similar notion of a difference representation was considered previously by Hurlbert [Hur94], however, it was considered cyclically and did not fix the first symbol.

The three different representations for k-subsets are illustrated in Table 1 for $\mathbf{S}_3(5)$. The sequence 1101011100 is a universal cycle for $\mathbf{S}_3(5)$ using the shorthand representation, and the sequence 1112122113 is a universal cycle for $\mathbf{S}_3(5)$ using the difference representation. There does not exist a universal cycle for $\mathbf{S}_3(5)$ using the standard representation.

Subset	Shorthand	Difference	Standard
{1,2,3}	1110 (0)	111	123, 132, ..., or 321
{1,2,4}	1101 (0)	112	124, 142, ..., or 421
{1,2,5}	1100 (1)	113	125, 152, ..., or 521
{1,3,4}	1011 (0)	121	134, 143, ..., or 431
{1,3,5}	1010 (1)	122	135, 153, ..., or 531
{1,4,5}	1001 (1)	131	145, 154, ..., or 541
{2,3,4}	0111 (0)	211	234, 243, ..., or 432
{2,3,5}	0110 (1)	212	235, 253, ..., or 532
{2,4,5}	0101 (1)	221	245, 254, ..., or 542
{3,4,5}	0011 (1)	311	345, 354, ..., or 543

Table 1 Illustrating different representations for the elements of $\mathbf{S}_3(5)$. For the shorthand representatives, the omitted redundant symbol is highlighted in parentheses.

Multisets. Like with k-subsets, the choice of representation is critical to constructing universal cycles for k-multisets. It is well known that $|\mathbf{M}_k(n)| = \binom{n+k-1}{k}$. Universal cycles for $\mathbf{M}_k(n)$ were first considered by Hurlbert, Johnson, and Zahl [HJZ09] using a *standard* string representation, where 112, 121, or 211 could represent the multiset $\{1,1,2\}$. Similar to k-subsets, they demonstrate that universal cycles for $\mathbf{M}_k(n)$ exists only if n divides $\binom{n+k-1}{k}$. Another method to represent a k-multiset is with a *frequency map*, which is a length-n string $f_1 f_2 \cdots f_n$ where each f_i represents the number of occurrences of i in the multiset. Note, $\sum_{i=1}^{n} f_i = k$. For instance, 110 is the frequency map representation for the multiset $\{1,2\}$ in $\mathbf{M}_2(3)$. It is a simple exercise to demonstrate that universal cycles for $\mathbf{M}_k(n)$ do not exist using this representation for $n,k \geq 2$. However, observe that the final symbol f_n in a frequency map is redundant: its value can be determined from the previous $n-1$ values, $f_n = k - \sum_{i=1}^{n-1} f_i$. Thus, we say $f_1 \cdots f_{n-1}$ is a *shorthand frequency* representation for a multiset in $\mathbf{M}_k(n)$. Observe that this set corresponds to all strings over the alphabet $\{0,1,\ldots,k\}$ of length $n-1$, where the weight of each string is bounded above by k.

[1] The application of this universal cycle to subsets was first noted in [BG11].

A *difference* representation can also be used for k-multisets. Consider the same definition applied for k-subsets except assign the first symbol of the difference representative to be *one less* than the smallest symbol in the multiset. Equivalently, apply the original definition of a difference representative directly, but define the k-multisets to be over the ground set $\{0, 1, \ldots, n-1\}$. Observe that by applying this difference representation, the set of k-multisets corresponds to all strings over the alphabet $\{0, 1, \ldots, n-1\}$ of length k, where the weight of each string is bounded above by $n-1$.

These three different representations for k-multisets are illustrated in Table 2 for $\mathbf{M}_3(3)$. The sequence 0011021203 is a universal cycle for $\mathbf{M}_3(3)$ using the shorthand frequency representation, and the sequence 0001011002 is a universal cycle for $\mathbf{M}_3(3)$ using the difference representation. There does not exist a universal cycle for $\mathbf{M}_3(3)$ using the standard representation.

Multiset	Shorthand frequency	Difference	Standard
{1, 1, 1}	30 (0)	000	111
{1, 1, 2}	21 (0)	001	112, 121, or 211
{1, 1, 3}	20 (1)	002	113, 131, or 311
{1, 2, 2}	12 (0)	010	122, 212, or 221
{1, 2, 3}	11 (1)	011	123, 132, 213, 231, 312, or 321
{1, 3, 3}	10 (2)	020	133, 313, or 331
{2, 2, 2}	03 (0)	100	222
{2, 2, 3}	02 (1)	101	223, 232, or 322
{2, 3, 3}	01 (2)	110	233, 323, or 332
{3, 3, 3}	00 (3)	200	333

Table 2 Illustrating different representations for the elements of $\mathbf{M}_3(3)$. For the shorthand frequency representatives, the omitted redundant symbol is highlighted in parentheses.

An application of universal cycles for k-multisets to proximity sensor networks is discussed in [CWLW24].

Main Results. Recall that the difference representatives of $\mathbf{S}_k(n)$ and both the shorthand frequency and difference representatives of $\mathbf{M}_k(n)$ are each special cases of fixed-length strings over a prescribed alphabet, where the strings have an upper bound on their weight. Thus, known results regarding bounded-weight de Bruijn sequences (see Section 3) can be applied to obtain the following main results of this paper.

1. We demonstrate an algorithm to construct a universal cycle $\mathcal{S}_1$ for $\mathbf{S}_k(n)$ using difference representatives that runs in $\mathcal{O}(1)$ amortized time per symbol. Moreover, given any subset in $\mathbf{S}_k(n)$ with difference representative $d_1 d_2 \cdots d_k$, the symbol following this string in $\mathcal{S}_1$ can be computed in $\mathcal{O}(n)$ time.
2. We demonstrate an algorithm to construct a universal cycle $\mathcal{M}_1$ for $\mathbf{M}_k(n)$ using shorthand frequency representatives that runs in $\mathcal{O}(1)$ amortized time per symbol. Moreover, given any subset in $\mathbf{M}_k(n)$ with shorthand frequency representative $f_1 f_2 \cdots f_{n-1}$, the symbol following this string in $\mathcal{M}_1$ can be computed in $\mathcal{O}(n)$ time.
3. We demonstrate an algorithm to construct a universal cycle $\mathcal{M}_2$ for $\mathbf{M}_k(n)$ using difference representatives that runs in $\mathcal{O}(1)$ amortized time per symbol. Moreover, given any subset in $\mathbf{M}_k(n)$ with difference representative $d_1 d_2 \cdots d_k$, the symbol following this string in $\mathcal{M}_2$ can be computed in $\mathcal{O}(n)$ time.

The last two results are the first known universal cycle constructions for k-multisets. All the algorithms require at most $\mathcal{O}(nt)$ space. Implementations for many known universal cycle constructions, including the algorithms presented in this paper, are available at `http://debruijnsequence.org` [dbs25].

Outline of paper. In Section 2 we present preliminary definitions, notation, and provide a brief background on the cycle-joining process and concatenation trees. In Section 3 we describe a known algorithm to construct

universal cycles for t-ary strings of length n with a given weight constraint and provide additional new insights into equivalent constructions. We also introduce a new construction with interesting properties. Then in Section 4, we apply these constructions to k-subsets and k-multisets.

2 Preliminaries

Let $\Sigma_t = \{0, 1, 2, \ldots, t-1\}$ and let $\Sigma_t(n)$ denote the set of all length-n strings over Σ_t. Recall, the *weight* of a string is the sum of its symbols. Let $\mathrm{weight}(\alpha)$ denote the weight of a string α. Let $\Sigma_t(n, w)$ denote the subset of all strings in $\Sigma_t(n)$ with weight *at most* w. Later in this paper we will be interested in strings as they are listed in *colex* order, which is standard lexicographic order when the strings are read from right to left. For instance, the following set of bounded-weight strings is listed in colex order:

$$\Sigma_3(3, 2) = \{000, 100, 200, 010, 110, 020, 001, 101, 011, 002\}.$$

Consider two strings α and β. Let $\alpha \cdot \beta$ denote the concatenation of α and β, and let β^j denote j copies of β concatenated together. The *aperiodic prefix* of α is the shortest string β such that $\alpha = \beta^j$ for some $j \geq 1$. A string is said to be *aperiodic* if it is the same as its aperiodic prefix; otherwise, it is said to be *periodic*.

A *necklace class* is an equivalence class of strings under rotation. A *necklace* is the lexicographically smallest string in a necklace class. Let $\mathbf{N}_t(n)$ denote the set of length-n necklaces over Σ_t. For example,

$$\mathbf{N}_3(3) = \{000, 001, 002, 011, 012, 021, 022, 111, 112, 122, 222\}.$$

Let $\mathbf{N}_t(n, w)$ denote the subset of all strings in $\mathbf{N}_t(n)$ with weight at most w. For example,

$$\mathbf{N}_3(3, 2) = \{000, 001, 002, 011\}.$$

A *feedback function* f maps strings from $\Sigma_t(n)$ to Σ_t. A *feedback shift register* is a function that maps a string $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ to $\mathtt{a}_2\cdots\mathtt{a}_n f(\alpha)$ for a given feedback function f. The *pure cycling register* (PCR) is a feedback shift register with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{a}_1$. It partitions any set closed under rotation into necklace classes that can be represented by necklaces from $\mathbf{N}_t(n)$. Consider a subset $\mathbf{S}$ of strings in $\Sigma_t(n+1)$ where each string has a unique shorthand representative like those outlined in Section 1. Let $\mathbf{S}'$ denote this set of shorthand representatives. Then each string $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n \in \mathbf{S}'$ corresponds to a unique string $\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n\mathtt{z} \in \mathbf{S}$. We call $\mathtt{z}$ the missing symbol. The *missing symbol register* (MSR) is a feedback shift register for such sets $\mathbf{S}'$ with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{z}$. The MSR partitions any shorthand set $\mathbf{S}'$ closed under rotation into necklace classes that can be represented by necklaces in $\mathbf{N}_t(n+1)$ (see Example 2 in Section 3.2). The MSR has been applied to shorthand representatives of permutations, subsets, and more generally, strings with fixed content in [SW23].

2.1 Cycle-joining trees and successor rules

In this section, assume that the representatives for a set of strings $\mathbf{S}$ is just $\mathbf{S}$ itself. Given a universal cycle $\mathcal{U} = \mathtt{u}_1\mathtt{u}_2\cdots\mathtt{u}_m$ for a set of strings $\mathbf{S}$, a *successor rule* for $\mathcal{U}$ is a function that maps each string α in $\mathbf{S}$ to the symbol following α in $\mathcal{U}$. Thus, starting with any string in $\mathbf{S}$, $\mathcal{U}$ (considered cyclically) can be constructed by repeatedly applying its successor rule.

Let $\mathcal{U}_i$ denote a universal cycle for $\mathbf{S}_i \subseteq \Sigma_t(n)$. Two universal cycles $\mathcal{U}_1$ and $\mathcal{U}_2$ are said to be *disjoint* if $\mathbf{S}_1 \cap \mathbf{S}_2 = \emptyset$. Let $\mathtt{x}, \mathtt{y}$ be distinct symbols in Σ_t. If $\sigma = \mathtt{x}\mathtt{s}_2\cdots\mathtt{s}_n$ and $\hat{\sigma} = \mathtt{y}\mathtt{s}_2\cdots\mathtt{s}_n$, then σ and $\hat{\sigma}$ are said to be *conjugates* of each other, and $(\sigma, \hat{\sigma})$ is called a *conjugate pair*. Two disjoint universal cycles $\mathcal{U}_1$ and $\mathcal{U}_2$ can be joined together to form a single universal cycle by swapping the successor of each string in a conjugate

pair $(\sigma, \hat{\sigma})$, where σ is found in $\mathcal{U}_1$ and $\hat{\sigma}$ is found in $\mathcal{U}_2$. This is the well-known *cycle-joining process* where the two cycles $\mathcal{U}_1$ and $\mathcal{U}_2$ are joined via the conjugate pair $(\sigma, \hat{\sigma})$. A *cycle-joining tree* $\mathbb{T}$ is an *unordered* tree where the nodes correspond to a disjoint set of universal cycles $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\ell$ and an edge between $\mathcal{U}_i$ and $\mathcal{U}_j$ is defined by a conjugate pair $(\sigma, \hat{\sigma})$ such that $\sigma \in \mathbf{S}_i$ and $\hat{\sigma} \in \mathbf{S}_j$. For our purposes, we consider cycle-joining trees to be rooted. In the binary case, repeatedly joining adjacent cycles in a cycle-joining tree together yields a *unique* universal cycle for $\mathbf{S}_1 \cup \mathbf{S}_2 \cup \cdots \cup \mathbf{S}_\ell$. For non-binary alphabets, different universal cycles are possible depending on the order that the cycles are joined [SSTW24]. Thus, the following property was introduced to produce a generic successor rule for non-binary alphabets based on the conjugate pairs.

Chain Property: If a node in a cycle-joining tree $\mathbb{T}$ has two children joined via conjugate pairs $(\mathtt{x}\mathtt{s}_2 \cdots \mathtt{s}_n, \mathtt{y}\mathtt{s}_2 \cdots \mathtt{s}_n)$ and $(\mathtt{x}'\mathtt{t}_2 \cdots \mathtt{t}_n, \mathtt{y}'\mathtt{t}_2 \cdots \mathtt{t}_n)$, then $\mathtt{s}_2 \cdots \mathtt{s}_n \neq \mathtt{t}_2 \cdots \mathtt{t}_n$.

Let $\mathbb{T}$ be a cycle-joining tree satisfying the Chain Property for an underlying set of strings $\mathbf{S}$, where the nodes are joined by a set $\mathbf{C}$ of conjugate pairs. If the tree contains ℓ cycles then $\mathbf{C}$ contains $\ell-1$ conjugate pairs, each corresponding to an edge in $\mathbb{T}$. Say γ *belongs to* a conjugate pair $(\sigma, \hat{\sigma})$ if either $\gamma = \sigma$ or $\gamma = \hat{\sigma}$. Let $C_1, C_2, \ldots, C_m$ denote a maximal length path of nodes in $\mathbb{T}$ such that for each $1 \leq i < m$, the node C_i is the parent of C_{i+1} and they are joined via a conjugate pair of the form $(\mathtt{x}_i\beta, \mathtt{x}_{i+1}\beta)$; β is the same in each conjugate pair. Call such a path a *chain* of length m, and define $g(\mathtt{x}_i\beta) = \mathtt{x}_{i+1}$ for $i < m$ and $g(\mathtt{x}_m\beta) = \mathtt{x}_1$ for each string belonging to a conjugate pair. Then the following function h is a successor rule for a corresponding universal cycle for $\mathbf{S}$, where $f(\alpha)$ is the underlying feedback function (such as the PCR or MSR) that induces the cycles in $\mathbb{T}$:

$$h(\alpha) = \begin{cases} g(\alpha) & \text{if } \alpha \text{ belongs to a conjugate pair in } \mathbf{C}; \\ f(\alpha) & \text{otherwise.} \end{cases}$$

This rule corresponds to the function $\uparrow f_1(\alpha)$ in [SSTW24] when f is the feedback function for the PCR.

Example 1 Consider the cycle-joining tree with nodes induced by the PCR illustrated in Figure 1. The path of nodes $\mathtt{000} \to \mathtt{001} \to \mathtt{002} \to \mathtt{003} \to \mathtt{004}$ form a chain where $\beta = 00$. The conjugate pairs are $(\mathtt{000}, \mathtt{100})$, $(\mathtt{100}, \mathtt{200})$, $(\mathtt{200}, \mathtt{300})$, and $(\mathtt{300}, \mathtt{400})$, respectively. Thus, $h(\mathtt{000}) = \mathtt{1}$, $h(\mathtt{100}) = \mathtt{2}$, $h(\mathtt{2000}) = \mathtt{3}$, $h(\mathtt{300}) = \mathtt{4}$, and $h(\mathtt{400}) = \mathtt{0}$.

Applying the rule directly requires storing all the conjugate pairs; however, in our application it is relatively straightforward to test whether a string belongs to a conjugate pair in $\mathcal{O}(n)$ time using only $\mathcal{O}(n)$ space. In this paper, we will define cycle-joining trees based on the PCR and MSR, and then apply $h(\alpha)$ to construct corresponding universal cycles efficiently.

2.2 Concatenation trees

In this section we present a simplified presentation of "left" concatenation trees as originally defined in [SSTW24], by adding an assumption that every non-root periodic node in a cycle-joining tree is a leaf.

A *bifurcated ordered tree* (BOT) is a rooted tree where each child of a node belongs to either the *left-children* or the *right-children*, and the nodes within each group are ordered. Let $\mathbb{T}$ be a PCR-based cycle-joining tree rooted at r satisfying the Chain Property. A *concatenation tree* based on $\mathbb{T}$ converts $\mathbb{T}$ into a BOT by ordering the children and assigning possibly new labels to represent each node (cycle). The parent-child relationship remains

the same and each node has a *change index*, which is the unique index where a node's label differs from that of its parent based on the conjugate pair joining the two nodes. This index is unique using our added assumption that all non-root periodic nodes are leaves, except for the case which occurs if a node is the child of a periodic root. We will handle that case as it arises. The root r can be assigned an arbitrary change index. If a node has change index c, all children with a change index less than or equal to c are classified as left children and are ordered from smallest to largest change index. All children with a change index greater than c are classified as right children and are also ordered from smallest to largest change index. An *RCL ordering* traverses a concatenation tree recursively from the root by first visiting all **R**ight children, then the **C**urrent node, then the **L**eft children. For example, see Figure 1 which illustrates a cycle-joining tree, a corresponding concatenation tree, and its RCL ordering.

Given a concatenation tree T, let $\mathcal{U}_T$ be the sequence obtained by concatenating the aperiodic prefixes of each node as they are visited in RCL order. The following result corresponds to Theorem 3 from [SSTW24].

▶ **Theorem 1.** *Let T be a concatenation tree for a PCR-based cycle-joining tree with the Chain Property for an underlying set* **S**. *Then $\mathcal{U}_T$ is a universal cycle for* **S** *with successor rule $h(\alpha)$.*

3 Bounded weight de Bruijn sequences

A universal cycle for $\Sigma_t(n)$ is known as a *de Bruijn sequence*. A universal cycle for the subsets of $\Sigma_t(n)$ where there is an upper or lower bound on the weight is called a *bounded weight de Bruijn sequence*. The lexicographically smallest binary de Bruijn sequence with a lower bound on the weight can be constructed in $\mathcal{O}(1)$ amortized time per symbol [SWW14]. The algorithm is generalized to non-binary alphabets in [SWW16] by providing an equivalent greedy construction. There are eight $\mathcal{O}(n)$ time successor-rule constructions for bounded weight de Bruijn sequences over an arbitrary alphabet given in [GSWW20]: four with a lower bound on the weight and four with an upper bound on the weight.[2] Each of the above algorithms requires $\mathcal{O}(n)$ space. We are interested in universal cycles for $\Sigma_t(n, w)$, which are strings with an upper bound of w on the weight.

In this section we focus on one of the successor rules from [GSWW20] that constructs a universal cycle for $\Sigma_t(n, w)$ with interesting properties. Since $\Sigma_t(n, w)$ is closed under rotation, the PCR partitions the set into necklace cycles. These cycles (nodes) can be represented by the necklaces in $\mathbf{N}_t(n, w)$. For example, the cycles for $\Sigma_5(3, 4)$ can be represented by the following necklaces as they are listed in colex order (this order will be useful later):

$$\mathbf{N}_5(3,4) = \{\mathtt{000}, \mathtt{001}, \mathtt{011}, \mathtt{111}, \mathtt{021}, \mathtt{031}, \mathtt{002}, \mathtt{012}, \mathtt{112}, \mathtt{022}, \mathtt{003}, \mathtt{013}, \mathtt{004}\}.$$

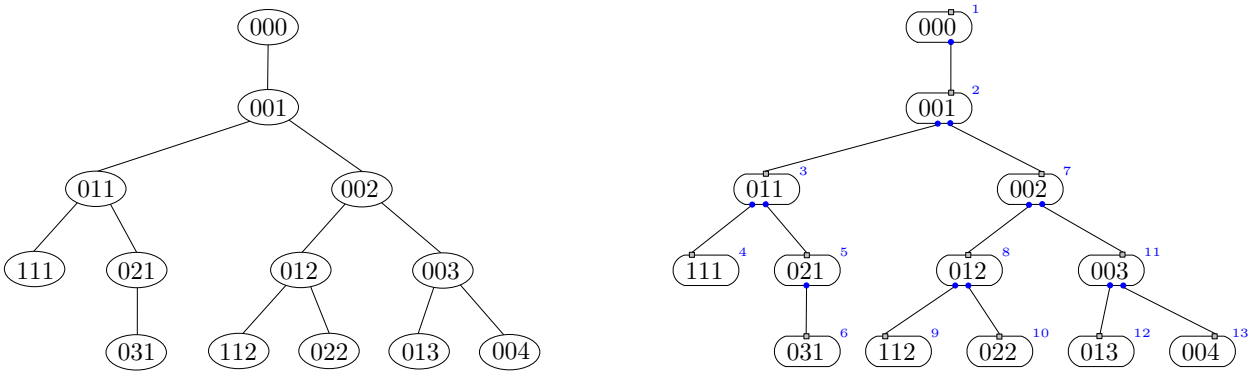
The following parent rule[3] defines a cycle-joining tree rooted at 0^n for the cycles represented by the necklaces in $\mathbf{N}_t(n, w)$.

> **First non-zero parent rule (with root 0^n)**: the parent of non-root node $\mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n$, where j denotes the smallest index such that $\mathtt{a}_j \neq 0$, is $0^{j-1}(\mathtt{a}_j - 1)\mathtt{a}_{j+1} \cdots \mathtt{a}_n$.

As an example, the cycle-joining tree for $\Sigma_5(3, 4)$ based on this *first non-zero* parent rule is shown in Figure 1.

[2] Naïvely, each successor rule requires up to t necklace tests, each requiring $\mathcal{O}(n)$ time. However applying optimizations similar to the upcoming proof of Theorem 6, these successor rules can be implemented with a constant number of necklace tests.

[3] This rule was called *first non-1* in [GSWW20] since the alphabet considered was $\{1, 2, \ldots, t\}$.



(a) Cycle-joining tree

(b) Concatenation tree

Figure 1 (a) The first non-zero cycle-joining tree for $\Sigma_5(3,4)$ where the nodes are represented by the necklaces in $\mathbf{N}_5(3,4)$. (b) A corresponding concatenation tree where the change index of the root is assigned to the final symbol, as indicated by the small box on top of the node. All children are left children, and the labels on the nodes do not need to change. An RCL traversal of this tree visits the necklaces in colex order.

▶ **Lemma 2.** *Let $\mathbb{T}$ be a PCR-based cycle-joining tree induced by the first non-zero parent rule and let $\alpha = \mathtt{0}^{i-1}\mathtt{x}\mathtt{a}_{i+1}\cdots\mathtt{a}_n$ and $\beta = \mathtt{0}^{j-1}\mathtt{y}\mathtt{b}_{j+1}\cdots\mathtt{b}_n$ be two distinct nodes (necklaces) in $\mathbb{T}$ where $i,j \leq n$ and $\mathtt{x},\mathtt{y} > 0$. Then the following hold.*

1. *If $\alpha \neq \mathtt{0}^{n-1}\mathtt{1}$ then the parent of α is an aperiodic necklace. (All non-root periodic nodes are leaves.)*
2. *$\mathbb{T}$ has the Chain Property.*
3. *If α and β are periodic then they have different parents.*

Proof. **(1)** Since α is a necklace, its prefix $\mathtt{a}_1\cdots\mathtt{a}_i = \mathtt{0}^{i-1}\mathtt{a}_i$ is lexicographically less than or equal to every other substring in α of the same length when α is considered cyclically. Its parent γ has prefix $\mathtt{0}^{i-1}(\mathtt{a}_i - 1)$ which is strictly less than all other substrings in γ considered cyclically, which means γ is a necklace. Furthermore, this also implies γ is aperiodic as long as $\gamma \neq \mathtt{0}^n$, which is the case only when $\alpha = \mathtt{0}^{n-1}\mathtt{1}$.

(2-3) Suppose α and β have the same parent γ. If $i = j$ then $\alpha = \beta$, a contradiction to α and β being distinct. Since they are necklaces $\mathtt{a}_n, \mathtt{b}_n > \mathtt{0}$. Without loss of generality, suppose $i < j$. The strings in the conjugate pair joining α and γ have suffix $\mathtt{a}_n\mathtt{0}^{i-1}$ and the strings in the conjugate pair joining β and γ have suffix $\mathtt{b}_n\mathtt{0}^{j-1}$. Thus, the Chain Property is satisfied. Furthermore, suppose α and β are both periodic. Since β is periodic, it must be that γ has a substring 0^{j-1} appearing after the j-th symbol. But this implies that α also has the same substring since α and β share the same parent. This contradicts the fact that α is a necklace. Thus α and β have different parents. ◀

The following successor rule based on the first non-zero cycle-joining tree corresponds to the successor rule g_4 in [GSWW20]. In its original presentation, the alphabet under consideration is $\{\mathtt{1},\mathtt{2},\ldots,\mathtt{t}\}$ whereas we state the rule for the alphabet $\Sigma_t = \{\mathtt{0},\mathtt{1},\ldots,\mathtt{t}-\mathtt{1}\}$. It is also a space-efficient implementation of the generic successor rule $h(\alpha)$ presented in Section 2.1 that does not require storing the set of conjugate pairs.

First non-zero (Grandmama) successor rule for $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$

Let j be the largest index of $\mathtt{a}_2\mathtt{a}_3\cdots\mathtt{a}_n$ such that $\mathtt{a}_j \neq \mathtt{0}$ or $j = 1$ if no such index exists. Let $\mathtt{x}$ be the largest symbol in $\{\mathtt{1},\mathtt{2},\ldots,\mathtt{t}-\mathtt{1}\}$ such that $\mathtt{0}^{n-j}\mathtt{x}\mathtt{a}_2\cdots\mathtt{a}_j$ is a necklace and $\text{weight}(\mathtt{x}\mathtt{a}_2\mathtt{a}_3\cdots\mathtt{a}_n) \leq w$,

or let $\mathtt{x} = -1$ if no such symbol exists.

$$h_1(\alpha) = \begin{cases} 0 & \text{if } \mathtt{x} \neq -1 \text{ and } \mathtt{a}_1 = \mathtt{x}; \\ \mathtt{a}_1{+}1 & \text{if } \mathtt{x} \neq -1 \text{ and } \mathtt{a}_1 < \mathtt{x}; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

Let $\mathcal{U}_t(n,w)$ denote the universal cycle for $\Sigma_t(n,w)$ obtained by starting with $\mathtt{0}^n$ and repeatedly applying the successor rule $|\Sigma_t(n,w)| - n$ times on the last n symbols to obtain the next symbol in the sequence. For example:

$$\mathcal{U}_5(3,4) = \underline{\mathtt{0 \cdot 00}}\mathtt{1 \cdot 011 \cdot 1 \cdot 021 \cdot 031 \cdot 002 \cdot 012 \cdot 112 \cdot 022 \cdot 003 \cdot 013 \cdot 004}.$$

Observe that this sequence has an interesting property: it corresponds to concatenating the aperiodic prefixes of the necklaces in $\mathbf{N}_5(3,4)$ as they appear in colex order. This is not surprising since when there is no bound on the weight, i.e., $w \geq n(t-1)$, the first non-zero successor rule is equivalent to the successor rule for the *Grandmama de Bruijn sequence* [DHS+18]. The Grandmama sequence can also be described by the following very simple concatenation scheme:

Concatenate together the aperiodic prefixes of the necklaces in $\mathbf{N}_t(n)$ as they appear in colex order.

Applying Theorem 1, this concatenation construction can be generalized to the set $\Sigma_t(n,w)$. Let $\mathbb{T}$ denote the cycle-joining tree for $\Sigma_t(n,w)$ defined by the first non-zero parent rule, where the cycles (nodes) are induced by the PCR – they correspond to the necklaces in $\mathbf{N}_t(n,w)$. From Lemma 2, $\mathbb{T}$ satisfies the Chain Property and all the non-root periodic nodes are leaves. Thus, let T be the concatenation tree derived from $\mathbb{T}$, where the change index of the root node $\mathtt{0}^n$ is the index of the rightmost symbol. Define the label of the single child of the root to be $\mathtt{0}^{n-1}\mathtt{1}$. This accounts for the one ambiguity in our simplified concatenation tree definition and this choice abides by the conditions from the original definition in [SSTW24]. Recall that the labels of all other nodes are defined recursively based on the label of their parent. It follows from Lemma 2, that the labels of all other nodes are necklaces.

▶ **Theorem 3.** *For $n, t \geq 1$ and $w \geq 0$, the universal cycle $\mathcal{U}_t(n,w)$ for $\Sigma_t(n,w)$ can be constructed by concatenating the aperiodic prefixes of the necklaces in $\mathbf{N}_t(n,w)$ as they appear in colex order.*

Proof. By Theorem 1, the sequence obtained by concatenating the aperiodic prefixes of the nodes in T as they appear RCL order has successor rule $h_1(\alpha)$, which generates $\mathcal{U}_t(n,w)$ starting with $\mathtt{0}^n$. Each child of a node is a left child. As a result, the RCL-ordering of T corresponds to a pre-order traversal of T. Since the symbol changed between a parent and child always increases from parent to child, this ordering will list the nodes (the necklaces in $\mathbf{N}_t(n,w)$) in colex order. ◀

By traversing the concatenation tree T in RCL order we can generate $\mathcal{U}_t(n,w)$. To do this efficiently, we dynamically compute the children of each node as we traverse T, starting from the root $\mathtt{0}^n$. Consider a node $\alpha = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n$ with change index c and weight w'. If $w' = w$, then α has no children. Let α_i denote the string α with the symbol at index i incremented by 1. If α_i is not a necklace for $1 < i \leq c$ then clearly α_{i-1} is also not a necklace. If $\mathtt{a}_c < k-1$, we test if α_c is a necklace. If it is not a necklace, then α has no children. Otherwise, we test α_{c-1}, then α_{c-2}, and so on until we find the largest index $i \leq c$ such that α_i is not a necklace or $i = 0$. The children of α are thus $\alpha_{i+1}, \cdots, \alpha_{c-1}$, including α_c if $\mathtt{a}_c < k-1$. Thus, starting from the root $\mathtt{0}^n$ we can dynamically create the tree T and visit its nodes in RCL order. Testing if a string is a necklace and also determining the length of the aperiodic prefix of a necklace can be computed in $\mathcal{O}(n)$ time [Boo80].

Each successful necklace test can be assigned to the corresponding child, and if there is a failed necklace test it can be assigned to the current node. Thus, the nodes can be visited in $\mathcal{O}(n)$ amortized time. A complete C implementation of this RCL traversal algorithm to construct $\mathcal{U}_t(n,w)$ is given in the Appendix.

▶ **Theorem 4.** *$\mathcal{U}_t(n,w)$ can be constructed in $\mathcal{O}(1)$ amortized time per symbol using $\mathcal{O}(nt)$ space.*

Proof. Recall from Lemma 2 that the parent of every non-root periodic necklace is both unique and aperiodic. Discounting the root, this means that the number of periodic necklaces is less than or equal to the number of aperiodic necklaces. Since the aperiodic prefix of each aperiodic necklace contains n symbols, this implies that $|\mathcal{U}_t(n,w)| \geq \frac{n}{2}|\mathbf{N}_t(n,w) - \{0^n\}|$. Since each necklace in the concatenation tree T is visited in $\mathcal{O}(n)$ amortized time, this implies that $\mathcal{U}_t(n,w)$ can be generated in $\mathcal{O}(1)$ amortized time per symbol. When visiting each node α, only a constant amount of extra memory is required if we update and restore the values in α to test if a given α_i is a necklace and to visit each child. Since the depth of the recursion is bounded by nt, this construction requires $\mathcal{O}(nt)$ space for the run-time stack. ◀

3.1 Fixed weight and weight range

Some applications may require both an upper and lower bound on the weights. Binary *fixed weight de Bruijn sequences* can be constructed by considering a shorthand representation [RSW12]. The shorthand strings correspond to all binary strings of length $n-1$ with weight in the range $[w-1, w]$. This result is applied in [SWW13] to construct binary *weight-range de Bruijn sequences* for binary strings with weight in an arbitrary range $[w_1, w_2]$. Both constructions generate the sequences in $\mathcal{O}(1)$ amortized time per symbol using $\mathcal{O}(n)$ space. There is no published construction for weight-range de Bruijn sequences over non-binary alphabets. As a special case, fixed-weight de Bruijn sequences for strings in $\Sigma_t(n)$ with weight exactly w using a shorthand representation correspond to weight-range de Bruijn sequences for strings in $\Sigma_t(n-1)$ with weight in the range $[max(0, w-t+1), w]$. If $w < t$, then such sequences correspond to bounded weight de Bruijn sequences with an upper bound on the weight of w.

▶ **Theorem 5.** *The universal cycle $\mathcal{U}_t(n,w)$ is a fixed-weight de Bruijn sequence using shorthand representatives when $w < t$. Moreover, the sequence $\mathcal{U}_t(n,w)$ with the 0^n substring replaced with 0^{n-1} is a fixed-weight de Bruijn sequence using shorthand representatives when $w = t$.*

3.2 Applying the MSR

In this section, we present another bounded weight de Bruijn sequence construction for $\Sigma_t(n,w)$ with the added constraint that $w < t$. Recall, the strings in $\Sigma_t(n,w)$ can be thought of as shorthand representatives for the subset of $\Sigma_t(n+1)$ containing strings with fixed weight $w < t$. Thus, when considering such a set $\Sigma_t(n,w)$, the missing symbol `z` for a given string in the set is well defined. Recall also that the MSR partitions $\Sigma_t(n,w)$ into equivalence classes corresponding to necklaces in $\mathbf{N}_t(n+1)$.

Example 2 Consider $\Sigma_5(3,4)$. The MSR partitions this set of 35 strings into the following 10 equivalence classes (columns):

000	001	010	002	011	020	003	012	021	111
004	013	103	022	112	202	031	121	211	
040	130	030	220	120		310	210	110	
400	300	301	200	201		100	101	102	

Each equivalence class (cycle) can be represented by a necklace in $\mathbf{N}_5(4)$ with weight $w = 4$. Each necklace is highlighted by reading the first symbol down each column.

Using the MSR, cycle-joining trees and successor rules can be created in a similar manner as we did when applying the PCR using the *first non-zero* parent rule. In this case, the nodes have length $n+1$. By decrementing the first non-zero symbol the following symbol in the parent necklace is incremented based on the MSR, maintaining the weight constraint.

First non-zero (MSR) (with root $\mathtt{0}^n\mathtt{w}$): the parent of non-root node $\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_{n+1}$, where j denotes the smallest index such that $\mathtt{a}_j \neq 0$, is $\mathtt{0}^{j-1}(\mathtt{a}_j{-}1)(\mathtt{a}_{j+1}{+}1)a_{j+2}\ldots\mathtt{a}_{n+1}$.

This parent rule is well-defined since the weight constraint implies that $j < n+1$ and $\mathtt{a}_{j+1} < t-1$. As an example, see the cycle-joining tree for $\Sigma_k(n,w)$ in Figure 2.

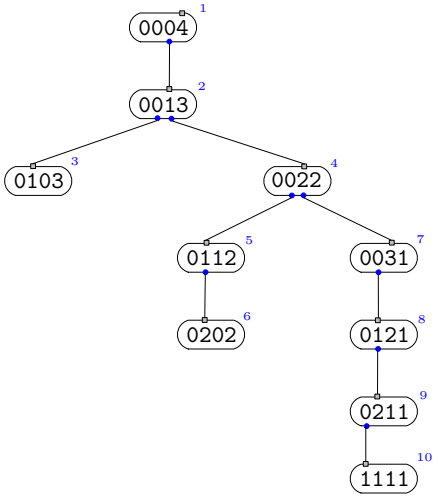


Figure 2 The first non-zero cycle-joining tree based on the MSR for $\Sigma_5(3,4)$ where the nodes are represented by the necklaces in $\mathbf{N}_5(4)$ with weight exactly $w = 4$. The labeled pre-order traversal of this tree visits the necklaces in *reverse* colex order. We draw the tree in the style of a concatenation tree even though they were not formally defined for MSR-based cycle-joining trees.

The following is a space-efficient successor rule based on the functions $g(\alpha)$ and $h(\alpha)$ (recall its definition in Section 2.1) for this cycle-joining tree, obtained by considering the corresponding conjugate pairs.

First non-zero (MSR) successor rule for $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ with missing symbol $\mathtt{z}$

Let j be the largest index of $\mathtt{a}_2\mathtt{a}_3\cdots\mathtt{a}_n$ such that $\mathtt{a}_j \neq \mathtt{0}$ or $j = 1$ if no such index exists. Let $\mathtt{x}$ be the largest symbol in $\{\mathtt{1},\mathtt{2},\ldots,\mathtt{t}{-}\mathtt{1}\}$ such that (i) $\mathtt{y} = \mathtt{z} - \mathtt{x} + \mathtt{a}_1$ is in Σ_t and (ii) $\mathtt{0}^{n-j}\mathtt{x}\mathtt{y}\mathtt{a}_2\cdots\mathtt{a}_j$ is a necklace, or let $\mathtt{x} = -1$ if no such symbol exists.

$$h_2(\alpha) = \begin{cases} 0 & \text{if } \mathtt{x} \neq -1 \text{ and } \mathtt{z} = \mathtt{x}; \\ \mathtt{z}{+}1 & \text{if } \mathtt{x} \neq -1 \text{ and } \mathtt{z} < \mathtt{x}; \\ \mathtt{z} & \text{otherwise.} \end{cases}$$

Let $\mathcal{V}_t(n,w)$ denote the universal cycle for $\Sigma_t(n,w)$ obtained by starting with $\mathtt{0}^n$ and repeatedly applying the above successor rule $|\Sigma_t(n,w)| - n$ times on the last n symbols to obtain the next symbol in the sequence. Naïvely, in order to determine $\mathtt{x}$ the function $h_2(\alpha)$ will require testing up to t strings to see whether they are a necklace. Testing whether a string is a necklace can be done in $\mathcal{O}(n)$ time [Boo80]. However, with some fairly straightforward optimizations, we can reduce the number of necklace tests to one.

▶ **Theorem 6.** *For $n, t \geq 2$ and $0 \leq w < t$, the universal cycle $\mathcal{V}_t(n,w)$ for $\Sigma_t(n,w)$ can be constructed via the successor rule $h_2(\alpha)$ in $\mathcal{O}(n)$ time per symbol using $\mathcal{O}(n)$ space.*

Proof. We demonstrate that the value $\mathtt{x}$ in the successor rule can be determined by performing at most one necklace test. Let $\beta = \mathtt{0}^{n-j}\mathtt{x}\mathtt{y}\mathtt{a}_2 \cdots \mathtt{a}_j$. Let $\mathtt{x}'$ denote the smallest symbol following a substring $\mathtt{0}^{n-j}$ in $\mathtt{a}_2 \cdots \mathtt{a}_j$, or let $\mathtt{x}' = \mathtt{t} - \mathtt{1}$ if no such substring exists. In order for β to be a necklace, clearly $\mathtt{x} \leq \mathtt{x}'$. Furthermore, $\mathtt{x} + \mathtt{y} = \mathtt{a}_1 + \mathtt{z}$ to maintain the weight constraint, which means $\mathtt{x} \leq \mathtt{a}_1 + \mathtt{z}$. Consider $\mathtt{x}$ to be the minimum of $\mathtt{x}'$ and $\mathtt{a}_1 + \mathtt{z}$. If this value is 0, then no necklace test is required. If β is a necklace for this value of $\mathtt{x}$, no more tests are required as it is the largest possible value for $\mathtt{x}$. Otherwise, decrementing the value of $\mathtt{x}$ (and incrementing $\mathtt{y}$) will result in a necklace since the length $n - j + 1$ prefix of β will be smaller than all other substrings of β (considered cyclically) of the same length, by the values of $\mathtt{x}'$ and the fact that $\mathtt{y} \neq 0$. ◀

The following result follows from the fact that the strings in $\Sigma_t(n,w)$ can be thought of as shorthand representatives for the subset of $\Sigma_t(n+1)$ containing strings with fixed weight $w < t$.

▶ **Theorem 7.** *The universal cycle $\mathcal{V}_t(n,w)$ is a fixed-weight de Bruijn sequence using shorthand representatives when $w < t$. Moreover, the sequence $\mathcal{V}_t(n,w)$ with the $\mathtt{0}^n$ substring replaced with $\mathtt{0}^{n-1}$ is a fixed-weight de Bruijn sequence using shorthand representatives when $w = t$.*

Interestingly, $\mathcal{V}_t(n,w)$ also appears to have interesting concatenation properties. Consider:

$$\mathcal{V}_5(3,4) = \underline{\mathtt{000}}\mathtt{4} \cdot \mathtt{0013} \cdot \mathtt{0103} \cdot \mathtt{0022} \cdot \mathtt{0112} \cdot \mathtt{02} \cdot \mathtt{0031} \cdot \mathtt{0121} \cdot \mathtt{0211} \cdot \mathtt{1}.$$

This sequence corresponds to the concatenation of the aperiodic prefixes of the necklaces in $\mathbf{N}_5(4)$ with exactly weight $w = 4$ as they appear in *reverse* colex order! Moreover, observe that a preorder traversal of the corresponding "concatenation-like tree" (see Figure 2) also corresponds to visiting the necklaces in reverse colex order. These observations lead to the following conjecture, where $\mathcal{V}'_t(n,w)$ denotes the sequence obtained by concatenating the aperiodic prefixes of the necklaces in $\mathbf{N}_t(n+1)$ with weight w as they appear in *reverse* colex order.

▶ **Conjecture 8.** *The universal cycle $\mathcal{V}_t(n,w)$ is equivalent to $\mathcal{V}'_t(n,w)$, where $n, t \geq 2$ and $w < t$. Moreover, the universal cycle can be generated in $\mathcal{O}(1)$ amortized time per symbol.*

We believe that this conjecture can be proved by generalizing the (PCR-based) concatenation tree framework [SSTW24], then following a similar analysis to the one we gave for the sequence $\mathcal{U}_t(n,w)$ to obtain the running time result.

4 Application to k-subsets and k-multisets

In this section, we apply the results from the previous section to k-subsets assuming $n \geq k \geq 1$, and k-multisets assuming $n, k \geq 2$. Recall the following observations made in Section 1.

- When the subsets of $\mathbf{S}_k(n)$ are represented by their difference representatives with each symbol $\mathtt{x}$ mapped to $\mathtt{x}-\mathtt{1}$, the subsets correspond to $\Sigma_{n-k+1}(k, n-k)$.

- When the multisets of $\mathbf{M}_k(n)$ are represented by their shorthand frequency representatives, the multisets correspond to $\Sigma_{k+1}(n-1,k)$.
- When the multisets of $\mathbf{M}_k(n)$ are represented by their difference representatives, the multisets correspond to $\Sigma_n(k,n-1)$.

These observations immediately give rise to the following three results.

▶ **Theorem 9.** *$\mathcal{U}_{n-k+1}(k,n-k)$ and $\mathcal{V}_{n-k+1}(k,n-k)$ are universal cycles for $\mathbf{S}_k(n)$ using difference representatives, where each symbol* `x` *in the representative is mapped to* `x` $-$ `1`.

Example 3 Consider two universal cycles for $\Sigma_4(3,3)$:

$\mathcal{U}_4(3,3) = \texttt{0}\cdot\texttt{001}\cdot\texttt{011}\cdot\texttt{1}\cdot\texttt{021}\cdot\texttt{002}\cdot\texttt{012}\cdot\texttt{003}$, and

$\mathcal{V}_4(3,3) = \texttt{0003}\cdot\texttt{0012}\cdot\texttt{0021}\cdot\texttt{0111}\cdot\texttt{0201}$.

By mapping each symbol `x` to `x` $+$ `1` in the above sequences, we obtain universal cycles $\mathcal{S}_1$ and $\mathcal{S}_2$ for $\mathbf{S}_3(6)$ using difference representatives:

$\mathcal{S}_1 = \texttt{1}\cdot\texttt{112}\cdot\texttt{122}\cdot\texttt{2}\cdot\texttt{132}\cdot\texttt{113}\cdot\texttt{123}\cdot\texttt{114}$, and

$\mathcal{S}_2 = \texttt{1114}\cdot\texttt{1123}\cdot\texttt{1132}\cdot\texttt{1222}\cdot\texttt{1312}$.

Each universal cycle above has the expected length of $\binom{6}{3} = 20$.

▶ **Theorem 10.** *$\mathcal{U}_{k+1}(n-1,k)$ and $\mathcal{V}_{k+1}(n-1,k)$ are universal cycles for $\mathbf{M}_k(n)$ using shorthand frequency representatives.*

▶ **Theorem 11.** *$\mathcal{U}_n(k,n-1)$ and $\mathcal{V}_n(k,n-1)$ are universal cycles for $\mathbf{M}_k(n)$ using difference representatives.*

Example 4 The universal cycles $\mathcal{U}_5(3,4)$ and $\mathcal{V}_5(3,4)$ are universal cycles for $\mathbf{M}_4(4)$ using shorthand frequency representatives:

$\mathcal{U}_5(3,4) = \texttt{0}\cdot\texttt{001}\cdot\texttt{011}\cdot\texttt{1}\cdot\texttt{021}\cdot\texttt{031}\cdot\texttt{002}\cdot\texttt{012}\cdot\texttt{112}\cdot\texttt{022}\cdot\texttt{003}\cdot\texttt{013}\cdot\texttt{004}$, and

$\mathcal{V}_5(3,4) = \texttt{0004}\cdot\texttt{0013}\cdot\texttt{0103}\cdot\texttt{0022}\cdot\texttt{0112}\cdot\texttt{02}\cdot\texttt{0031}\cdot\texttt{0121}\cdot\texttt{0211}\cdot\texttt{1}$.

The universal cycles $\mathcal{U}_4(4,3)$ and $\mathcal{V}_4(4,3)$ are universal cycles for $\mathbf{M}_4(4)$ using difference representatives:

$\mathcal{U}_4(4,3) = \texttt{0}\cdot\texttt{0001}\cdot\texttt{01}\cdot\texttt{0011}\cdot\texttt{0111}\cdot\texttt{0021}\cdot\texttt{0002}\cdot\texttt{0102}\cdot\texttt{0012}\cdot\texttt{0003}$, and

$\mathcal{V}_4(4,3) = \texttt{00003}\cdot\texttt{00012}\cdot\texttt{00102}\cdot\texttt{00021}\cdot\texttt{00111}\cdot\texttt{01011}\cdot\texttt{00201}$.

Each universal cycle above has the expected length of $\binom{4+4-1}{4} = 35$.

The above three results imply that universal cycles for k-subsets and k-multisets can be constructed in $\mathcal{O}(1)$ amortized time per symbol, or via an $\mathcal{O}(n)$ time per symbol successor rule. The latter two results are the first known universal cycle constructions for k-multisets.

5 Acknowledgment

Joe Sawada (grant RGPIN-2025-03961) gratefully acknowledges research support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

BG11 A. Blanca and A. P. Godbole. On universal cycles for new classes of combinatorial structures. *SIAM Journal on Discrete Mathematics*, 25(4):1832–1842, 2011.

Boo80 K. S. Booth. Lexicographically least circular substrings. *Information Processing Letters*, 10(4/5):240–242, 1980.

CDG92 F. Chung, P. Diaconis, and R. Graham. Universal cycles for combinatorial structures. *Discrete Mathematics*, 110(1):43–59, 1992.

CHHM09 D. Curtis, T. Hines, G. Hurlbert, and T. Moyer. Near-universal cycles for subsets exist. *SIAM Journal on Discrete Mathematics*, 23(3):1441–9, 2009.

CWLW24 C. S. Chen, W. S. Wong, Y.-H. Lo, and T.-L. Wong. Multiset combinatorial Gray codes with application to proximity sensor networks, 2024.

dbs25 De Bruijn sequence and universal cycle constructions (2025). http://debruijnsequence.org., 2025.

DHS+18 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.

DL16 M. Debski and Z. Lonc. Universal cycle packings and coverings for k-subsets of an n-set. *Graphs and Combinatorics*, 32(6):2323–2337, Nov 2016.

GJKO20 S. Glock, F. Joos, D. Kühn, and D. Osthus. Euler tours in hypergraphs. *Combinatorica*, 40(5):679–690, November 2020.

GSWW20 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.

HJZ09 G. Hurlbert, T. Johnson, and J. Zahl. On universal cycles for multisets. *Discrete Mathematics*, 309(17):5321–5327, 2009. Generalisations of de Bruijn Cycles and Gray Codes/Graph Asymmetries/Hamiltonicity Problem for Vertex-Transitive (Cayley) Graphs.

Hur94 G. Hurlbert. On universal cycles for k-subsets of an n-set. *SIAM Journal on Discrete Mathematics*, 7(4):598–604, 1994.

Jac93 B. Jackson. Universal cycles of k-subsets and k-permutations. *Discrete Mathematics*, 117:141–150, 07 1993.

JSH09 B. Jackson, B. Stevens, and G. Hurlbert. Research problems on Gray codes and universal cycles. *Discrete Mathematics*, 309(17):5341–5348, 2009.

Lan12 M. Lanius. *Universal Cycles for k-subsets of an n-set*. Honors thesis, Wellesley College, 2012.

RSW12 F. Ruskey, J. Sawada, and A. Williams. De Bruijn sequences for fixed-weight binary strings. *SIAM Journal on Discrete Mathematics*, 26(2):605–617, 2012.

Rud13 Y. Rudoy. An inductive approach to constructing universal cycles on the k-subsets of $[n]$. *The Electronic Journal of Combinatorics*, 20:P18, 2013.

SBE+02 B. Stevens, P. Buskell, P. Ecimovic, C. Ivanescu, A. Malik, A. Savu, T. Vassilev, H. Verrall, B. Yang, and Z. Zhao. Solution of an outstanding conjecture: The non-existence of universal cycles with $k = n - 2$. *Discrete Mathematics*, 258:193–204, Dec 2002.

SSTW24 J. Sawada, J. Sears, A. Trautrim, and A. Williams. Concatenation trees: A framework for efficient universal cycle and de Bruijn sequence constructions. *arXiv preprint arXiv:2308.12405*, 2024.

SW13 J. Sawada and A. Williams. A Gray code for fixed-density necklaces and Lyndon words in constant amortized time. *Theoretical Computer Science*, 502:46–54, 2013. Generation of Combinatorial Structures.

SW23 J. Sawada and A. Williams. Constructing the first (and coolest) fixed-content universal cycle. *Algorithmica*, 85(6):1754–1785, Jun 2023.

SWW13 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, LNCS 8288*, pages 388–401, 2013.

SWW14 J. Sawada, A. Williams, and D. Wong. The lexicographically smallest universal cycle for binary strings with minimum specified weight. *Journal of Discrete Algorithms*, 28:31–40, 2014. StringMasters 2012, 2013 Special Issue (Volume 1).

SWW16 J. Sawada, A. Williams, and D. Wong. Generalizing the classic greedy and necklace constructions of de Bruijn sequences and universal cycles. *Electron. J. Combin.*, 23(1):Paper 1.24, 20, 2016.

A C implementation of the bounded weight “Grandmama” de Bruijn sequence

```
//=========================================================================
// Genereate the (upper) bounded weight "Grandmama" de Bruijn sequence
// over alphabet {0,1, ... , t-1} and upper bound on weight of w.
// It can be applied to construct a UC for k-subsets or k-multisets of [n].
//=========================================================================
#include<stdio.h>
int a[100],n,t,w;

//=========================================================================
// Test if a[1..n] is a necklace, if so return the length of its aperiodic
// prefix; otherwise return 0
//=========================================================================
int IsNecklace(int a[]) {
    int i,p=1;

    for (i=1; i<=n; i++) {
        if (a[i] < a[i-p]) return 0;
        if (a[i] > a[i-p]) p=i;
    }
    if (n%p == 0) return p;
    return 0;
}
//=========================================================================
// Visit "first non-zero" concatenation tree in RCL order by dynamically
// generating children of the current node a[1..n] with aperiodic prefix of
// length p, change index c, and weight w2.
// This visits necklaces with bounded weight w in colex order.
//=========================================================================
void RCL(int a[], int p, int c, int w2) {
    int i,j;

    // VISIT: Print aperiodic prefix of a[1..n]
    for (i=1; i<=p; i++) printf("%d", a[i]);
    if (w2 == w) return;     // No children when max weight achieved

    // Scan from c left to determine the first index for a child
    a[c]++;
    if (a[c] < t && !IsNecklace(a))  { a[c]--; return; }
    a[c]--;

    j=c-1;
    a[j] = 1;
    while (j >=1 && IsNecklace(a)) {
        a[j] = 0; j--; a[j] = 1;
    }
    a[j] = 0;

    // Visit children from left to right, handle change index separately
    for (i=j+1; i<c; i++) {
        a[i] = 1;
        RCL(a, IsNecklace(a), i, w2+1);
        a[i] = 0;
    }
    if (a[c] < t-1) {
        a[c]++;
        RCL(a, IsNecklace(a), c, w2+1);
        a[c]--;
    }
}
//=================================================
```

```
int main() {

    printf("Enter n t w: ");  scanf("%d %d %d", &n, &t, &w);
    for (int i=1; i<=n; i++) a[i] = 0;
    RCL(a,1,n,0);
}
```