Concatenation trees: A framework for efficient universal cycle and de Bruijn sequence constructions

Joe Sawada
University of Guelph, Canada

Jackson Sears
Ontario Tech University, Canada

Andrew Trautrim
University of Guelph, Canada

Aaron Williams
Williams College, USA

Abstract

Classic cycle-joining techniques have found widespread application in creating universal cycles for a diverse range of combinatorial objects, such as shorthand permutations, weak orders, orientable sequences, and various subsets of k-ary strings, including de Bruijn sequences. In the most favorable scenarios, these algorithms operate with a space complexity of $O(n)$ and require $O(n)$ time to generate each symbol in the sequences. In contrast, concatenation-based methods have been developed for a limited selection of universal cycles. In each of these instances, the universal cycles can be generated far more efficiently, with an amortized time complexity of $O(1)$ per symbol, while still using $O(n)$ space. This paper introduces *concatenation trees*, which serve as the fundamental structures needed to bridge the gap between cycle-joining constructions and corresponding concatenation-based approaches. They immediately demystify the relationship between the classic Lyndon word (necklace) concatenation construction of de Bruijn sequences and a corresponding cycle-joining based construction. To underscore their significance, concatenation trees are applied to construct universal cycles for shorthand permutations, weak orders, and orientable sequences in $O(1)$-amortized time per symbol.

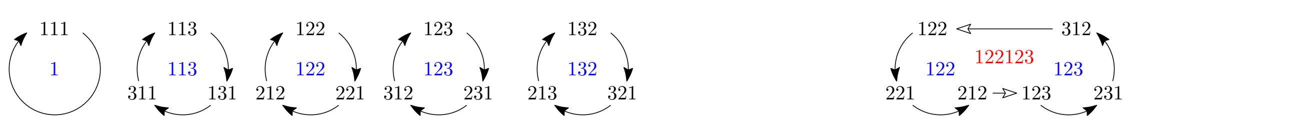
1 Introduction

Readers are likely familiar with the concept of a *de Bruijn sequence* (DB sequence), which is a circular string of length k^n in which every k-ary string of length n appears once as a substring. For example, a binary DB sequence for $n = 4$ is 0000100110101111. The study of these sequences dates back to Pingala's *Chandaḥśāstra* छन्दःशास्त्र ('A Treatise on Prosody') over two thousand years ago (see [33, 48, 49, 50]). They have a wide variety of well-known modern-day applications [1] and their theory is even being applied to de novo assembly of read sequences into a genome [4, 9, 35, 47, 52]. More broadly, when the underlying objects are not k-ary strings, the analogous concept is often called a *universal cycle* [8], and they have been studied for many fundamental objects including permutations [27, 32, 38, 51], combinations [10, 29, 30], set partitions [26], and graphs [6].

In this paper, we develop a concatenation framework for the generation of DB sequences and universal cycles. We prove that each such sequence is equivalent to one generated by a corresponding successor rule that is based on an underlying cycle-joining tree. As we demonstrate, the concatenation constructions can often be implemented to generate the sequences in $O(1)$-amortized time per symbol, whereas the corresponding successor-rule generally requires $O(n)$ time. To illustrate our approach, it is helpful to consider a slightly more complex object. A *weak order* is a way competitors can rank in an event, where ties are allowed. For example, in a horse race with five horses labeled h_1, h_2, h_3, h_4, h_5, the weak order (using a rank representation) 22451 indicates h_5 finished first, the horses h_1 and h_2 tied for second, horse h_3 finished fourth, and horse h_4 finished fifth. No horse finished third as a result of the tie for second. Let $\mathbf{W}(n)$ denote the set of weak orders of order n. For example, the thirteen weak orders for $n = 3$ are given below:

$$\mathbf{W}(3) = \{111, 113, 131, 311, 122, 212, 221, 123, 132, 213, 231, 312, 321\}.$$

Note that $\mathbf{W}(n)$ is closed under rotation. For this reason, we can apply the pure cycling register (PCR), which corresponds to the function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{a}_2\cdots\mathtt{a}_n\mathtt{a}_1$, to induce small cycles. Then, we repeatedly join the smaller cycles together to obtain a universal cycle. In this approach, we partition $\mathbf{W}(n)$ into equivalence classes under rotation. These classes are called *necklaces* and we use the lexicographically smallest member of each class as its representative. So $\{113, 131, 311\}$ is one class with representative 113, and $\{111\}$ is another class. Each class of size t has a universal cycle of length t, namely the representative's aperiodic prefix (i.e., the shortest prefix of a string that can be concatenated some number of times to create the entire string). So 113 is a universal cycle for $\{113, 131, 311\}$, and 1 is a universal cycle for $\{111\}$ (since 1 is viewed cyclically). Each necklace class can be viewed as a directed cycle induced by the PCR, where each edge corresponds to a rotation (i.e., the leftmost symbol is shifted out and then shifted back in as the new rightmost symbol), as seen in Figure 1a for $n = 3$. Two cycles can be joined together via a conjugate pair (formally defined in Section 2.1) to create a larger cycle as illustrated in Figure 1b. This is done by replacing a pair of rotation edges with a pair of edges that shift in a new symbol. Repeating this process yields a universal cycle 1113213122123 for $\mathbf{W}(3)$.



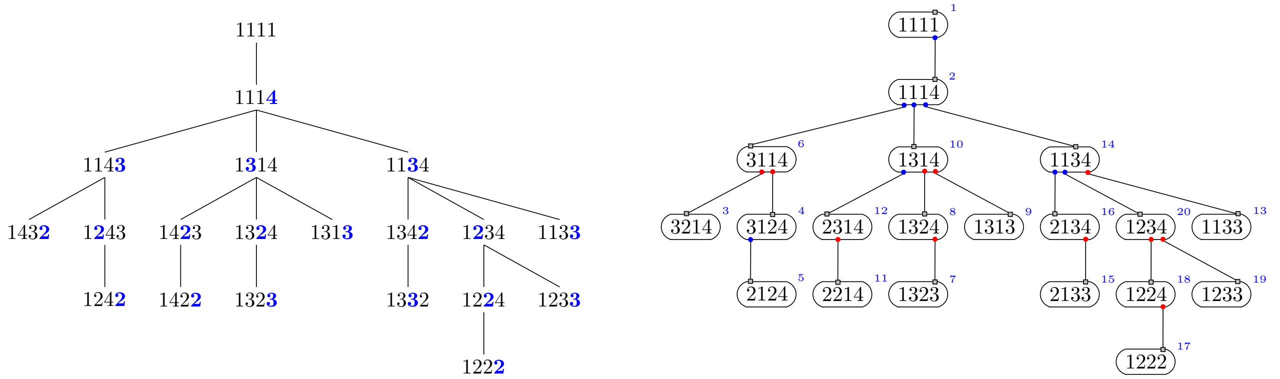
(a) Necklace cycles for $\mathbf{W}(3)$, where the representative of each class is at the top of its cycle, and the universal cycle is in the middle.

(b) Necklace cycles 122 and 123 are joined into a single cycle. The universal cycle for these strings is 122123.

Figure 1 Initial steps to building a universal cycle for $\mathbf{W}_3$.

In many cases, pairs of cycles can be joined together to form a cycle-joining tree. For example, Figure 2a illustrates a cycle-joining tree for $\mathbf{W}(4)$ based on an explicit parent rule stated in Section 5.3. Given a cycle-joining tree, existing results in the literature [23, 24] allow us to generate a corresponding universal cycle *one symbol at a time*. But what if we want to generate the universal cycle faster? For instance, suppose that instead of generating one symbol at a time, we can generate necklaces one at a time.[1] How can we do this? This goal of generating one necklace at a time has been achieved in only a

[1] In practice, a DB sequence (or universal cycle) does not need to be returned to an application one symbol at a time, but rather a word can be shared between the generation algorithm and the application. The algorithm repeatedly informs the application that the next batch of symbols in



(a) A cycle-joining tree for weak orders when $n = 4$. The precise parent rule appears in Section 5.3.

(b) A concatenation tree T_{weak} for weak orders when $n = 4$ illustrating the RCL order.

Figure 2 Two tree structures for creating a universal cycle for $\mathbf{W}_4$.

handful of cases [12, 17, 20, 37, 41]. Most notably, the DB sequence known as the Ford sequence, or the *Granddaddy* (see Knuth [34]), can be created by concatenating the associated representatives in lexicographic order [18], matching the DB sequence given earlier: 0 0001 0011 01 0111 1. But these concatenation constructions have been the exception rather than the rule, and there has been no theoretical framework for understanding why they work. Here, we provide the missing link. For example, the unordered cycle joining tree in Figure 2a is redrawn in Figure 2b. The new diagram is a bifurcated ordered tree (formally defined in Section 3), meaning that children are ordered and partitioned into left and right classes, and importantly some representatives have changed. If the tree is explored using an *RCL traversal* (i.e., right children, then current, then left children), then — presto! — a concatenation construction of a universal cycle for $\mathbf{W}(4)$ is created:

1 1114 3214 3124 2124 3114 1323 1324 13 1314 2214 2314 1133 1134 2133 2134 1222 1224 1233 1234.

Main result: This paper introduces *concatenation trees* and *RCL traversals*, which bridge the gap between k-ary PCR-based cycle-joining trees and concatenation constructions for corresponding universal cycles. We apply the framework to construct universal cycles in $O(1)$-amortized time per symbol using polynomial space for (1) shorthand permutations, (2) weak orders, (3) orientable sequences, and (4) DB sequences.

Our main result generalizes many interesting results for DB sequences and their relatives, with details provided in Section 2.2 and Section 5.1.

1. It demystifies the relationship between the successor rule and the concatenation construction of the previously mentioned Granddaddy DB sequence [16, 18], by providing a clear correspondence between the concatenation construction and the successor rule derived from an underlying cycle-joining tree.
2. Similar to the Grandaddy, it demystifies the relationship between the known successor rule and concatenation construction of the Grandmama DB sequence [12].
3. It provides the first proof of an observed correspondence between a successor rule construction [31, 45] and a simple concatenation construction observed in [20] (that we later name the *Granny* DB sequence).
4. It generalizes known results for bounded weight universal cycles [40, 43, 44] and universal cycles with forbidden 0^j substring [20, 44]; the latter has recent application in quantum key distribution schemes [7].

Additionally, we apply the framework to other combinatorial objects to highlight its general significance.

---

the sequence is ready. This allows the generation algorithm to slightly modify the shared word and provide $O(n)$ symbols to the application as efficiently as $O(1)$-amortized time [12].

1. Concatenation trees are applied to a $O(n)$ time per symbol cycle-joining construction for shorthand permutations [24] to generate the same universal cycle in $O(1)$-amortized time per symbol using $O(n^2)$ space.
2. Concatenation trees are applied to a $O(n)$ time per symbol cycle-joining construction for weak orders [46] to generate the same universal cycle in $O(1)$-amortized time per symbol using $O(n^2)$ space.
3. Concatenation trees are applied to a $O(n)$ time per symbol cycle-joining construction for orientable sequences [22] to generate the same universal cycle in $O(1)$-amortized time per symbol using $O(n^2)$ space.

While our focus is on PCR-based cycle-joining trees, preliminary evidence indicates that our framework can be generalized (though non-trivially) to other underlying feedback functions in the binary case including:

- the Complementing Cycling Register (CCR) with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = 1 \oplus \mathtt{a}_1 = \overline{\mathtt{a}}_1$,
- the Pure Summing Register (PSR) with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{a}_1 \oplus \mathtt{a}_2 \cdots \oplus \mathtt{a}_n$,
- the Complementing Summing Register (CSR) with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = 1 \oplus \mathtt{a}_1 \oplus \mathtt{a}_2 \cdots \oplus \mathtt{a}_n$, and
- the Pure Run-length Register (PRR) with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{a}_1 \oplus \mathtt{a}_2 \oplus \mathtt{a}_n$,

where $\oplus$ is addition modulo 2, and $\overline{\mathtt{x}}$ is the complement of $\mathtt{x}$. This has the potential to unify a large body of independent results, enabling new and interesting results. In particular, the recently introduced pure run-length register (PRR) [39] is conjectured to be the underlying feedback function used in a lexicographic composition construction [17]. Furthermore, the PRR is proved to be the underlying function used in the greedy prefer-same [13] and prefer-opposite [3] constructions; however, no concatenation construction is known. The first successor rule based on the complementing cycling register (CCR) is noted to have a very good local 0-1 balance [28]; however, no corresponding concatenation construction is known. There are two known CCR-based concatenation constructions [19, 20], but there is no clear correlation to an underlying cycle-joining approach, even though one appears to be equivalent to a successor rule from [23]. The cool-lex concatenation constructions [37] have equivalent underlying successor rules based on the pure summing register (PSR) and the complementing summing register (CSR). This correspondence was not observed until considering larger alphabets [41], though little insight to the correspondence is provided in the proof. Cycle-joining constructions based on the PSR/CSR are also considered in [14, 15].

Outline. In Section 2, we present the necessary background definitions and notation along with a detailed discussion of cycle-joining trees and their corresponding successor rules. In Section 3, we introduce bifurcated ordered trees, which are the structure underlying concatenation trees. In Section 4, we introduce concatenation trees along with a statement of our main result. In Section 5, we apply our framework to a wide variety of interesting combinatorial objects, including DB sequences. Implementation of the universal cycle algorithms presented in this paper are available at `http://debruijnsequence.org`.

2 Preliminaries

Let $\Sigma = \{0, 1, 2, \ldots, k-1\}$ denote an alphabet with k symbols. Let Σ^n denote the set of all length-n strings over Σ. Let $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ denote a string in Σ^n. The notation α^t denotes t copies of α concatenated together. The *aperiodic prefix* of α is the shortest string β such that $\alpha = \beta^t$ for some $t \geq 1$; the *period* of α is $|\beta|$. Let $\mathrm{ap}(\alpha_1, \alpha_2, \ldots, \alpha_n)$ denote the concatenation of the aperiodic prefixes of $\alpha_1, \alpha_2, \ldots, \alpha_t$. For example $\mathrm{ap}(0000, 0111, 1010) = 0011110$, and $\mathrm{ap}(010101) = 01$. Note that 010101 has period equal to 2. If the period of α is n, then α is said to be *aperiodic* (or primitive); otherwise, it is said to be *periodic* (or a proper power). When $k = 2$, let $\overline{\mathtt{a}}_i$ denote the complement of a bit $\mathtt{a}_i$.

A *necklace class* is an equivalence class of strings under rotation. A *necklace* is the lexicographically smallest representative of a necklace class. A *Lyndon word* is an aperiodic necklace. Let $\mathbf{N}_k(n)$ denote the set of all k-ary necklaces of order n. As an example, the six binary necklaces for $n = 4$ are: $\mathbf{N}_2(4) = \{0000, 0001, 0011, 0101, 0111, 1111\}$. Let $[\alpha]$ denote the set of all strings in α's necklace class, i.e., the set of all rotations of α. For example, $[0001] = [1000] = \{0001, 0010, 0100, 1000\}$ and $[0101] = \{0101, 1010\}$. The *pure cycling register* (PCR) is a shift register with feedback function $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{a}_1$. Starting with α, it induces a cycle containing the strings in α's necklace class. For example,

$$0001 \rightarrow 0010 \rightarrow 0100 \rightarrow 1000 \rightarrow \mathit{0001}$$

is a cycle induced by the PCR that can be represented by any string in the cycle. Given a tree T with nodes (cycles induced by the PCR) labeled by necklace representatives $\{\alpha_1, \alpha_2, \ldots, \alpha_t\}$, let $\mathbf{S}_T = [\alpha_1] \cup [\alpha_2] \cup \cdots \cup [\alpha_t]$. For example, if $n = 4$ and T contains two nodes $\{0001, 0101\}$ then $\mathbf{S}_T = \{0001, 0010, 0100, 1000\} \cup \{0101, 1010\}$.

Given $\mathbf{S} \subseteq \Sigma^n$, a *universal cycle* U for $\mathbf{S}$ is a cyclic sequence of length $|\mathbf{S}|$ that contains each string in $\mathbf{S}$ as a substring (exactly once). Given a universal cycle U for a set $\mathbf{S} \subseteq \Sigma^n$, a *successor rule* for U is a function $f : \mathbf{S} \to \Sigma$ such that $f(\alpha)$ is the symbol following α in U.

2.1 Cycle joining trees

In this section we review how two universal cycles can be joined to obtain a larger universal cycle. Let $\mathtt{x}, \mathtt{y}$ be distinct symbols in Σ. If $\alpha = \mathtt{x}\mathtt{a}_2 \cdots \mathtt{a}_n$ and $\hat{\alpha} = \mathtt{y}\mathtt{a}_2 \cdots \mathtt{a}_n$, then α and $\hat{\alpha}$ are said to be *conjugates* of each other, and $(\alpha, \hat{\alpha})$ is called a *conjugate pair*. The following well-known result (see for instance Lemma 3 in [42]) based on conjugate pairs is the crux of the cycle-joining approach.[2]

Theorem 1. *Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be disjoint subsets of Σ^n such that $\alpha = \mathtt{x}\mathtt{a}_2 \cdots \mathtt{a}_n \in \mathbf{S}_1$ and $\hat{\alpha} = \mathtt{y}\mathtt{a}_2 \cdots \mathtt{a}_n \in \mathbf{S}_2$; $(\alpha, \hat{\alpha})$ is a conjugate pair. If U_1 is a universal cycle for $\mathbf{S}_1$ with suffix α and U_2 is a universal cycle for $\mathbf{S}_2$ with suffix $\hat{\alpha}$ then $U = U_1U_2$ is a universal cycle for $\mathbf{S}_1 \cup \mathbf{S}_2$.*

Let U_i denote a universal cycle for $\mathbf{S}_i \subseteq \Sigma^n$. Two universal cycles U_1 and U_2 are said to be *disjoint* if $\mathbf{S}_1 \cap \mathbf{S}_2 = \emptyset$. A *cycle-joining tree* T is an unordered tree where the nodes correspond to a disjoint set of universal cycles $U_1, U_2, \ldots, U_t$; an edge between U_i and U_j is defined by a conjugate pair $(\alpha, \hat{\alpha})$ such that $\alpha \in \mathbf{S}_i$ and $\hat{\alpha} \in \mathbf{S}_j$. For our purposes, we consider cycle-joining trees to be rooted. If the cycles are induced by the PCR, i.e., the cycles correspond to necklace classes, then T is said to be a *PCR-based cycle-joining tree*. As examples, four PCR-based cycle-joining trees are illustrated in Figure 3; their nodes are labeled by the necklaces $\mathbf{N}_2(6)$. They are defined by the following *parent-rules*, which determines the parent of a given non-root node.

Four "simple" parent rules defining binary PCR-based cycle-joining trees

- T_1: rooted at 1^n and the parent of every other node $\alpha \in \mathbf{N}_2(n)$ is obtained by flipping the last 0.
- T_2: rooted at 0^n and the parent of every other node $\alpha \in \mathbf{N}_2(n)$ is obtained by flipping the first 1.
- T_3: rooted at 0^n and the parent of every other node $\alpha \in \mathbf{N}_2(n)$ is obtained by flipping the last 1.
- T_4: rooted at 1^n and the parent of every other node $\alpha \in \mathbf{N}_2(n)$ is obtained by flipping the first 0.

Note that for T_3 and T_4, the parent of a node α is obtained by first flipping the named bit and then rotating the string to its lexicographically least rotation to obtain a necklace. Each node α and its parent β are joined by a conjugate pair where the highlighted bit in α is the first bit in one of the conjugates. For example, the nodes $\alpha = 011011$ and $\beta = 001011$ in T_2 from Figure 3 are joined by the conjugate pair $(1\underline{10110}, 0\underline{10110})$.

When two adjacent nodes U_i and U_j in a cycle-joining tree T are joined to obtain U via Theorem 1 (rotating the cycles as appropriate), the nodes are unified and replaced with U (the edge between U_i and U_j is contracted). Repeating this process until only one node remains produces a universal cycle for $\mathbf{S}_1 \cup \mathbf{S}_2 \cup \cdots \cup \mathbf{S}_t$. In the binary case, the same universal cycle is produced, no matter the order in which the cycles are joined. This is because no string can belong to more than one conjugate pair in the underlying definition of T. However, when $k > 2$, the order that the cycles are joined can be important.

Example 1 The following illustrates two different ways to join the cycles in a PCR-based cycle-joining tree T for $n = 3$ and $k = 3$ with three nodes represented by $001, 002$, and 003 joined via conjugate pairs $(100, 200), (200, 300)$. Note the string 200 belongs to both conjugate pairs.

[2] The cycle-joining approach has graph theoretic underpinnings related to Hierholzer's algorithm for constructing Euler cycles [25].

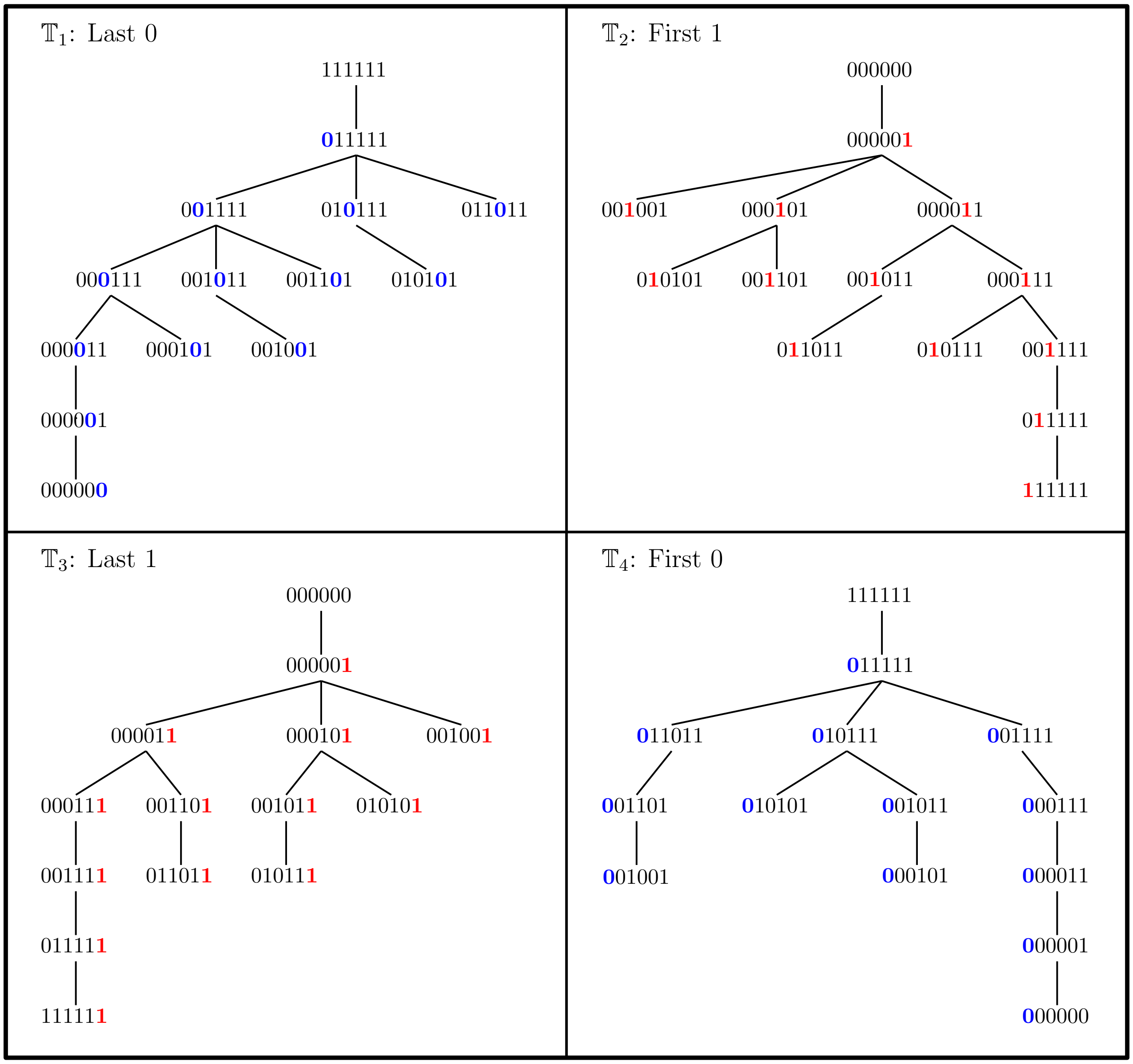
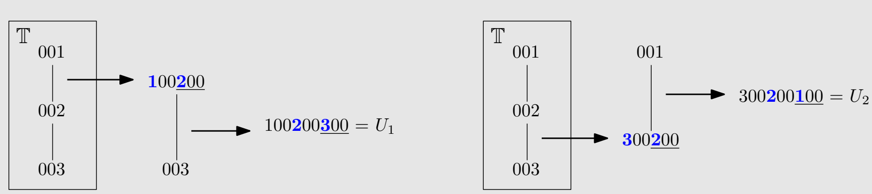


Figure 3 Cycle-joining trees for $n = 6$ and $k = 2$ derived from the four simple parent rules. The node 001101 is joined to a different parent cycle in each tree. In particular, the edge 001101–001111 in T_1 is obtained by flipping its last 0.

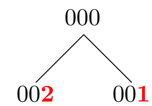


The resulting universal cycle for $\mathbf{S}_{\mathsf{T}} = [001] \quad [002] \quad [003]$ is different in each case.

In upcoming discussion regarding both successor rules and concatenation trees, we require the underlying cycle-joining trees to have the following property when $k > 2$.

Chain Property: If a node in a cycle-joining tree $\mathtt{T}$ has two children joined via conjugate pairs $(\mathtt{x}\mathtt{a}_2\cdots\mathtt{a}_n, \mathtt{y}\mathtt{a}_2\cdots\mathtt{a}_n)$ and $(\mathtt{x}\,\mathtt{b}_2\cdots\mathtt{b}_n, \mathtt{y}\,\mathtt{b}_2\cdots\mathtt{b}_n)$, then $\mathtt{a}_2\cdots\mathtt{a}_n \neq \mathtt{b}_2\cdots\mathtt{b}_n$.

Observe that the Chain Property is satisfied in Example 1, and is always satisfied when $k = 2$. The cycle-joining tree on the right with conjugate pairs ($0\underline{00}$, $1\underline{00}$) and ($0\underline{00}$, $2\underline{00}$) illustrates a rooted tree that does not satisfy the Chain Property.



2.2 Successor-rule constructions

Let $\mathtt{T}$ be a PCR-based cycle-joining tree where the nodes are joined by a set $\mathbf{C}$ of conjugate pairs. We say γ *belongs to* a conjugate pair $(\alpha, \hat{\alpha})$ if either $\gamma = \alpha$ or $\gamma = \hat{\alpha}$. If $k = 2$, the following function f_0 is a successor rule for the corresponding universal cycle for $\mathbf{S}_{\mathtt{T}}$ [23], where $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$:

$$f_0(\alpha) = \begin{cases} \overline{\mathtt{a}}_1 & \text{if } \alpha \text{ belongs to some conjugate pair in } \mathbf{C}; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

Applying the successor rule f_0 directly requires an exponential amount of memory to store the conjugate pairs. However, a cycle-joining tree defined by a straightforward parent rule may allow for a much more efficient implementation, using as little as $O(n)$ space and $O(n)$ time. Recall the four parent rules stated for the trees $\mathtt{T}_1$, $\mathtt{T}_2$, $\mathtt{T}_3$, $\mathtt{T}_4$. The upcoming four successor rules $\mathrm{pcr}_1, \mathrm{pcr}_2, \mathrm{pcr}_3, \mathrm{pcr}_4$, which correspond to f_0, are stated generally for any subtree T of the corresponding cycle-joining tree; they will be revisited in Section 5.1. Previously, these successor rules were stated for the entire trees in [23], and then for subtrees that included all nodes up to a given level [24] putting a restriction on the minimum or maximum weight (number of 1s) of any length-n substring.

<u>$\mathtt{T}_1$ (Last 0)</u> Let j be the smallest index of $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ such that $\mathtt{a}_j = 0$ and $j > 1$, or $j = n{+}1$ if no such index exists. Let $\gamma = \mathtt{a}_j\mathtt{a}_{j+1}\cdots\mathtt{a}_n0\mathtt{a}_2\cdots\mathtt{a}_{j-1} = \mathtt{a}_j\mathtt{a}_{j+1}\cdots\mathtt{a}_n01^{j-2}$.

$$\mathrm{pcr}_1(\alpha) = \begin{cases} \overline{\mathtt{a}}_1 & \text{if } \gamma \text{ is a necklace and } \mathtt{a}_2\cdots\mathtt{a}_n\overline{\mathtt{a}}_1 \quad \mathbf{S}_T; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

<u>$\mathtt{T}_2$ (First 1)</u> Let j be the largest index of $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ such that $\mathtt{a}_j = 1$, or $j = 0$ if no such index exists. Let $\gamma = \mathtt{a}_{j+1}\mathtt{a}_{j+2}\cdots\mathtt{a}_n1\mathtt{a}_2\cdots\mathtt{a}_j = 0^{n-j}1\mathtt{a}_2\cdots\mathtt{a}_j$.

$$\mathrm{pcr}_2(\alpha) = \begin{cases} \overline{\mathtt{a}}_1 & \text{if } \gamma \text{ is a necklace and } \mathtt{a}_2\cdots\mathtt{a}_n\overline{\mathtt{a}}_1 \quad \mathbf{S}_T; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

<u>$\mathtt{T}_3$ (Last 1)</u> Let $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ and let $\gamma = \mathtt{a}_2\mathtt{a}_3\cdots\mathtt{a}_n1$.

$$\mathrm{pcr}_3(\alpha) = \begin{cases} \overline{\mathtt{a}}_1 & \text{if } \gamma \text{ is a necklace and } \mathtt{a}_2\cdots\mathtt{a}_n\overline{\mathtt{a}}_1 \quad \mathbf{S}_T; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

<u>$\mathtt{T}_4$ (First 0)</u> Let $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ and let $\gamma = 0\mathtt{a}_2\mathtt{a}_3\cdots\mathtt{a}_n$.

$$\mathrm{pcr}_4(\alpha) = \begin{cases} \overline{\mathtt{a}}_1 & \text{if } \gamma \text{ is a necklace and } \mathtt{a}_2\cdots\mathtt{a}_n\overline{\mathtt{a}}_1 \quad \mathbf{S}_T; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

The DB sequences obtained by applying the four successor rules for $n = 6$ and $k = 2$ to $T = \mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3, \mathtt{T}_4$, respectively, are provided in Table 1. The spacing between some symbols are used to illustrate the correspondence to upcoming concatenation constructions. The DB sequence generated by pcr_1 is the well-known Ford sequence [16], and is

Successor rule	DB sequence for $n = 6$ and $k = 2$
pcr_1	0 000001 000011 000101 000111 001 001011 001101 001111 01 010111 011 011111 1
pcr_2	0 000001 001 000101 01 001101 000011 001011 011 000111 010111 001111 011111 1
pcr_3	1 111110 111100 111000 110 110100 110000 101110 101100 10 101000 100 100000 0
pcr_4	1 111110 110 100 100110 111010 10 110010 100010 111100 111000 110000 100000 0

Table 1 DB sequences resulting from the successor rules corresponding to the cycle-joining trees T_1, T_2, T_3, T_4 from Figure 3.

called the *Granddaddy* by Knuth [34]. It is the lexicographically smallest DB sequence, and it can also be generated by a prefer-0 greedy approach attributed to Martin [36]. Furthermore, Fredricksen and Maiorana [18] demonstrate an equivalent necklace (or Lyndon word) concatenation construction that can generate the sequence in $O(1)$-amortized time per bit. The DB sequence generated by pcr_2 is called the *Grandmama* by Dragon et al. [12]; it can also be generated in $O(1)$-amortized time per bit by concatenating necklaces in co-lexicographic order. The DB sequence generated by pcr_3, was first discovered by Jansen [31] for $k = 2$, then generalized in [45]. It is conjectured to have a concatenation construction by Gabric and Sawada [20], a fact we prove in Section 5.1. The DB sequence generated by pcr_4, was first discovered by Gabric, Sawada, Williams, and Wong [23]. No concatenation construction for this sequence was previously known which served as the initial motivation for this work.

2.2.1 Non-binary alphabets

Consider a non-binary alphabet where $k > 2$. Recall from Example 1, that the order the cycles are joined in a cycle-joining tree T may be important. This means defining a natural and generic successor rule is more challenging, especially if T does not satisfy the Chain Property, i.e., T has a node with two children joined via conjugate pairs of the form $(\mathtt{x}\beta, \mathtt{y}\beta)$ and $(\mathtt{x}\beta, \mathtt{z}\beta)$, for some k-ary string β. Thus, going forward, assume that T satisfies the Chain Property.

Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ denote a maximal length path of nodes in T such that for each $1 \leq i < m$, the node α_i is the parent of α_{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. We call such a path a *chain* of length m, and define $\text{first}(\mathtt{x}_i\beta) = \mathtt{x}_1$. For each such chain in T, assign a permutation $d_1d_2\cdots d_m$ of $\{1, 2, \ldots, m\}$ in which no element appears in its original position (a derangement).

Let $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$. If $\alpha = \mathtt{x}_i\beta$ belongs to a conjugate pair that joins two nodes in a chain $\alpha_1, \alpha_2, \ldots, \alpha_m$ with corresponding derangement $d_1d_2\cdots d_m$, let $g(\alpha) = \mathtt{x}_{d_i}$. Then the following function f_1 is a successor rule for a corresponding universal cycle for $\mathbf{S}_\mathsf{T}$ (based on the theory in [24]):

$$f_1(\alpha) = \begin{cases} g(\alpha) & \text{if } \alpha \text{ belongs to a conjugate pair in } \mathbf{C}; \\ \mathtt{a}_1 & \text{otherwise.} \end{cases}$$

When $k = 2$, $f_1 = f_0$.

Example 2 Continuing Example 1, let $\alpha = 300$; it belongs to a conjugate pair. Note that $\alpha_1 = 001$, $\alpha_2 = 002$, and $\alpha_3 = 003$ form a chain of length $m = 3$. If the derangement assigned to this chain is 231, then f_1 is the successor rule for the universal cycle 100200300. If the derangement assigned to this chain is 312, then f_1 is the successor rule for the universal cycle 300200100.

Perhaps the most natural derangements for the chains in T are of the form $23\cdots m1$ and $m12\cdots(m{-}1)$. Specifically, let:

- $\uparrow f_1(\alpha)$ denote the function $f_1(\alpha)$ when all chain derangements have the form $23\cdots m1$, and
- $\downarrow f_1(\alpha)$ denote the function $f_1(\alpha)$ when all chain derangements have the form $m12\cdots(m{-}1)$.

These are precisely the successor rules that correspond to our upcoming concatenation tree results. They are also the ones used in the generic successor rules stated in Theorem 2.8 and Theorem 2.9 from [24]; they lead to the definition of natural successor rules for eight different DB sequences including the k-ary Granddaddy (lex smallest) [18] and the k-ary Grandmama [12].

2.3 Insights into concatenation trees

The sequence in Table 1 generated by pcr_1 starting with 0^n has an interesting property: It corresponds to concatenating the *aperiodic prefixes* of each node in the corresponding cycle-joining tree T_1 (illustrated in Figure 3) as they are visited in post-order, where the children of a node are listed in lexicographic order. Notice also, that a post-order traversal visits the necklaces (nodes) as they appear in lexicographic order; this corresponds to the well-known Granddaddy necklace concatenation construction for DB sequences [18]. Similarly, the sequence generated by the successor rule pcr_2 starting with 0^n corresponds to concatenating the aperiodic prefixes of each node in the corresponding cycle-joining tree T_2 as they are visited in pre-order, where the children of a node are listed in colex order. This traversal visits the necklaces (nodes) as they appear in colex order, which is known as the Grandmama concatenation construction for DB sequences [12]. Unfortunately, this *magic* does not carry over to the trees T_3 and T_4, no matter how we order the children; the existing proofs for T_1 and T_2 offer no higher-level insights or pathways towards generalization.

Our discovery to finding a concatenation construction for a given successor rule is to tweak the corresponding cycle-joining tree by: (i) determining the appropriate representative of each cycle, (ii) determining an ordering of the children, and (iii) determining how the tree is traversed. The resulting concatenation trees for T_1, T_2, T_3, and T_4, which are formally defined in Section 4, are illustrated in Figure 8 for $n = 6$. The concatenation trees derived from T_1 and T_2 look very similar to the original cycle-joining trees. For the concatenation tree derived from T_3, the representatives are obtained by rotating the initial prefix of 0s of a necklace to the suffix; a post-order traversal produces the corresponding DB sequence in Table 1. This traversal corresponds to visiting these representatives in reverse lexicographic order that is equivalent to a construction defined in [20]. The concatenation tree derived from T_4 is non-trivial and proved to be the basis for discovering our more general result. Each representative is determined from its parent, and the tree differentiates "left-children" (blue dots) from "right-children" (red dots). A concatenation construction corresponding to pcr_4 is obtained by a somewhat unconventional traversal that recursively visits right-children, followed by the current node, followed by the left-children.

3 Bifurcated ordered trees

Our new "concatenation-tree" approach to generating universal cycles and DB sequences relies on tree structures that mix together ordered trees and binary trees. First we review basic tree concepts. Then we introduce our notion of a bifurcated ordered tree together with a traversal called an RCL traversal.

An *ordered tree* is a rooted tree in which the children of each node are given a total order. For example, a node in an ordered tree with three children has a first child, a second child, and a third (last) child. In contrast, a *cardinal tree* is a rooted tree in which the children of each node occupy specific positions. In particular, a *k-ary tree* has k positions for the children of each node. For example, each child of a node in a 3-ary tree is either a left-child, a middle child, or a right-child.

We consider a new type of tree that is both ordinal and cardinal; while ordered trees have one "type" of child, our trees will have two types of children. We refer to such a tree as a *bifurcated ordered tree* (*BOT*), with the two types of children being *left-children* and *right-children*. To illustrate bifurcated ordered trees, Figure 4 provides all BOTs with $n = 3$ nodes. This

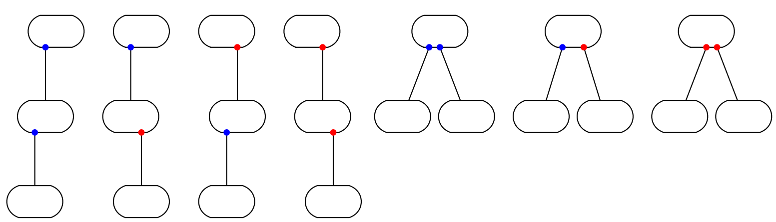


Figure 4 All eight bifurcated ordered trees (BOTs) with n=3 nodes. Each left-child descends from a blue •, while each right-child descends from a red •.

type of "ordinal-cardinal" tree seems quite natural and it is very likely to have been used in previous academic investigations. Nevertheless, the authors have not been able to find an exact match in the literature. In particular, 2-tuplet trees use a different notion of a root, and correspond more closely to ordered forests of BOTs. The number of BOTS with $n = 1, 2, \ldots, 12$ nodes is given by:

$$1, 2, 7, 30, 143, 728, 3876, 21318, 120175, 690690, 4032015, 23841480.$$

This listing corresponds to sequence A006013 in the Online Encyclopedia of Integer Sequences [2].

3.1 Right-Current-Left (RCL) traversals

The distinction between left-children and right-children in a BOT allows for a very natural notion of an *in-order traversal*: visit the left-children from first to last, then the current node, then the right-children from first to last. During our work with concatenation trees (see Section 4) it will be more natural to use a modified traversal, in which the right-children are visited before the left-children. Formally, we recursively define a *Right-Current-Left (RCL) traversal* of a bifurcated ordered tree starting from the root as follows:

- visit all right-children of the current node from first to last;
- visit the current node;
- visit all left-children of the current node from first to last.

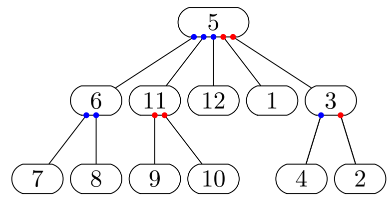


Figure 5 A BOT with its n=12 nodes labeled as they appear in RCL order.

Note that the resulting RCL order is not the same as a *reverse in-order traversal* (i.e., an in-order traversal written in reverse), since the children of each type are visited in the usual order (i.e., first to last) rather than in reverse order (i.e., last to first). An example of an RCL traversal is shown in Figure 5.

Define the following relationships given a node x in a BOT.

- A *right-descendant* of x is a node obtained by traversing down zero or more right-children.
- A *left-descendant* of x is a node obtained by traversing down zero or more left-children.
- The *rightmost left-descendant* of x is the node obtained by repeatedly traversing down the last left-child as long as one exists.
- The *leftmost right-descendant* of x is the node obtained by repeatedly traversing down the first right-child as long as one exists.

Note that a node is its own leftmost right-descendent if it has no right-children. Similarly, a node is its own rightmost left-descendent if it has no left-children. The following remark details the cases for when two nodes from a BOT appear consecutively in RCL order; they are illustrated in Figure 6.

Remark 2. If a bifurcated ordered tree has RCL traversal $\ldots, x, y, \ldots$, then one of the following three cases holds:

(a) x is an ancestor of y: y is the leftmost right-descendant of x's first left-child;
(b) x is a descendant of y: x is the rightmost left-descendent of y's last right-child;
(c) x and y are descendants of a common ancestor a (other than x and y): x is the rightmost left-descendant and y is the leftmost right-descendant of consecutive left-children or right-children of a.

Moreover, if the traversal sequence is cyclic (i.e., x is last in the ordering and y is first), there are three additional cases:

(d) x is an ancestor of y: x is the root and y is its leftmost right-descendant;
(e) x is a descendant of y: y is the root and x is its rightmost left-descendant;
(f) x and y are descendants of a common ancestor a (other than x and y): x is the rightmost left-descendant of the root, and y is the leftmost right-descendant of the root.

Figure 6 illustrates the six cases from the above remark. The three cases provided for cyclic sequences are stated in a way to convince the reader that all options are considered; however, they can be collapsed to the single case (f) if we allow the common ancestor a to be x or y.

4 Concatenation trees

Let T be a PCR-based cycle-joining tree rooted at r satisfying the Chain Property. In this section we describe how T can be converted into a labeled BOT $\mathcal{T}$ we call a *concatenation tree*. The nodes and the parent-child relationship in $\mathcal{T}$ are the same as in T; however, the labels (representatives) of the nodes may change. The definitions of these labels are defined recursively along with a corresponding *change index*, the unique index where a node's label differs from that of its parent. The root of

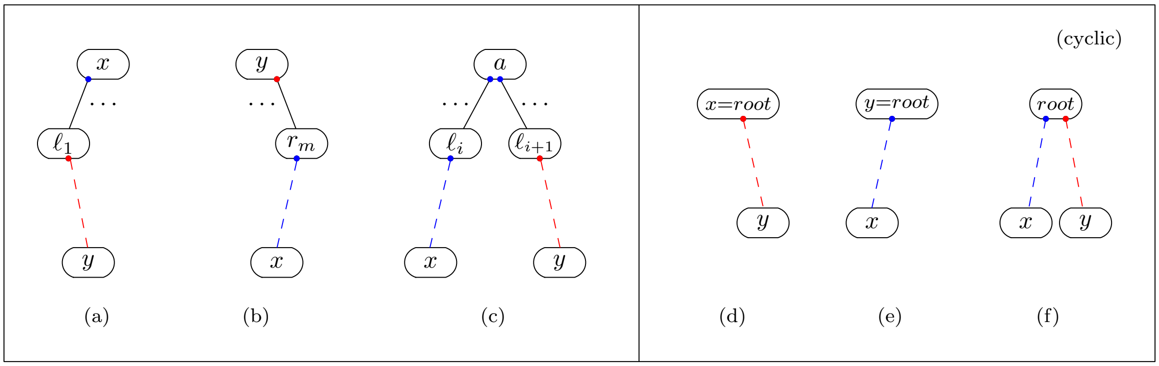


Figure 6 Illustrating the six cases outlined in Remark 2 for when y follows x in an RCL traversal. The final three cases hold when the traversal sequence is considered to be cyclic (i.e., x comes last and y comes first). In these images, ℓ_i and r_i refer to the ith left and right-child of their parent, respectively, and r_m refers to the last right-child of its parent. Dashed lines indicate leftmost right-descendants (red) and rightmost left-descendants (blue).

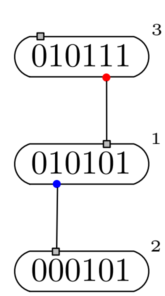
$\mathcal{T}$ is r, and it is assigned an arbitrary change index c.[3] The label of a non-root node γ depends on the label of its parent $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$, which can be written as $\beta_1\mathtt{x}\beta_2$ where $(\mathtt{x}\beta_2\beta_1, \mathtt{y}\beta_2\beta_1)$ is the conjugate pair joining α and γ in $\mathtt{T}$. If α is aperiodic, there is only one possible index i for $\mathtt{x}$; however, if it is periodic, there will be multiple such indices possible. If $\alpha = (\mathtt{a}_1\cdots\mathtt{a}_p)^q$ has period p with change index c where $jp < c \leq jp+p$ for some integer $0 \leq j < n/p$, then we say the *acceptable range* of α is $\{jp{+}1, \ldots, jp{+}p\}$. Note, if α is aperiodic, its acceptable range is $\{1, 2, \ldots, n\}$. Now, $\alpha = \beta_1\mathtt{x}\beta_2$ can be written uniquely such that $\mathtt{x}$ is found at an index i in α's acceptable range. The label of γ is defined to be $\beta_1\mathtt{y}\beta_2$ with change index i.

Example 3 Let $x = 001001001$ be the parent of $y = 001002001$ in a PCR-based cycle-joining tree $\mathtt{T}$ joined via the conjugate pair $(100100100, 200100100)$. Let α and γ denote the corresponding nodes in the concatenation tree $\mathcal{T}$. Suppose $\alpha = 100100100$ (a rotation of x) with change index 8. Since α has period $p = 3$, its acceptable range is $\{7, 8, 9\}$. Thus, $\beta_1 = 100100$, $\mathtt{x} = 1$, $\beta_2 = 00$, $\alpha = \beta_1\mathtt{x}\beta_2$, and $\gamma = 100100\mathbf{2}00$ (a rotation of y) with change index 7.

To complete the definition of $\mathcal{T}$, we must specify how the children of a node with change index c are partitioned into ordered left-children and right-children: The left-children are those with change index less than c, and the right-children are those with change index greater than c. Both are ordered by increasing change index. A child with change index c can be considered to be either a left-child or right-child. We say $\mathcal{T}$ is a *left concatenation tree* if every node that has the same change index as its parent is considered to be a left-child; $\mathcal{T}$ is a *right concatenation tree* if every node that has the same change index as its parent is considered to be a right-child. Let $\text{concat}(\mathtt{T}, c, \textit{left})$ denote the left concatenation tree derived from $\mathtt{T}$ and let $\text{concat}(\mathtt{T}, c, \textit{right})$ denote the right concatenation tree derived from $\mathtt{T}$, where in each case the root is assigned change index c. See Figure 7 for example concatenation trees, where the small gray box on top of each node indicates the node's change index.

Let $\text{RCL}(\mathcal{T}) = \text{ap}(\alpha_1, \alpha_2, \ldots, \alpha_t)$, where $\alpha_1, \alpha_2, \ldots, \alpha_t$ is the sequence of nodes visited in an RCL traversal of the concatenation tree $\mathcal{T}$. For example, if $\mathcal{T}$ is the *right* concatenation tree in Figure 7, then:

$$\text{RCL}(\mathcal{T}) = 011311\ 311\ 011210\ 011211\ 411211\ 211\ 011114\ 011111.$$



It is critical how we defined the acceptable range for periodic nodes, since our goal is to demonstrate that $\text{RCL}(\mathcal{T})$ produces a universal cycle. For example, consider three necklace class representatives (a) 010111, (b) 010101, and (c) 000101 where $n = 6$. They can be joined by flipping the last 0 in (b) and flipping the second 0 in (c); (a) is the parent of (b) and (b) is the parent of (c). A BOT for this cycle-joining tree is shown on the right. It is *not* a concatenation tree since the change index for the bottom node is outside the acceptable range of its periodic parent. Observe that $\text{ap}(010101, 000101, 010111) = 01000101010111$. Since the substring 010101 appears twice, it is not a universal cycle.

[3] Though the change index of the root is arbitrary, its choice may impact the "niceness" of the upcoming RCL sequence.

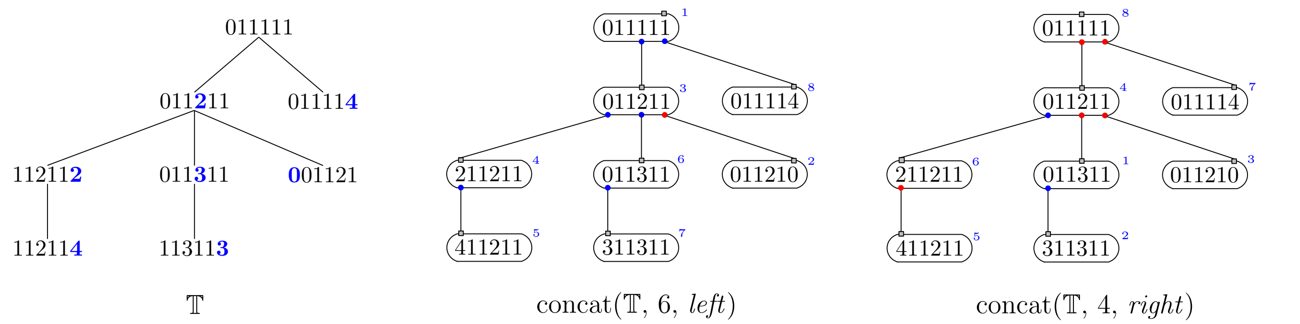


Figure 7 Left and right concatenation trees for a given cycle-joining tree T. The small blue numbers indicate the RCL order.

The concatenation trees for the four cycle-joining trees in Figure 3 are given in Figure 8. The only concatenation tree with both left-children and right-children is the one corresponding to $\text{concat}(\mathsf{T}_4, 6, \textit{left})$. In fact, it was the discovery of this tree that lead us to the introduction of BOTs and our definition of concatenation trees. We are now ready to state our main result.

> I **Theorem 3.** *Let* T *be a PCR-based cycle-joining tree satisfying the Chain Property. Let* $\mathcal{T}_1 = \text{concat}(\mathsf{T}, c, \textit{left})$ *and let* $\mathcal{T}_2 = \text{concat}(\mathsf{T}, c, \textit{right})$*. Then*
>
> - $\text{RCL}(\mathcal{T}_1\ )$ *is a universal cycle for* $\mathbf{S}_\mathsf{T}$ *with successor rule* $\uparrow f_1$*, and*
> - $\text{RCL}(\mathcal{T}_2\ )$ *is a universal cycle for* $\mathbf{S}_\mathsf{T}$ *with successor rule* $\downarrow f_1$*.*

Proof. Let $\mathcal{T}$ represent either $\mathcal{T}_1$ or $\mathcal{T}_2$. We specify whether $\mathcal{T}$ is a left-concatenation tree $\mathcal{T}_1$ or a right-concatenation tree $\mathcal{T}_2$ only when necessary. Let $\alpha_1, \alpha_2, \ldots, \alpha_t$ be the nodes of $\mathcal{T}$ as they are visited in RCL order. The proof of Theorem 3 is by induction on t. In the base case case when $t = 1$, the result is immediate; $\mathcal{T}$ contains a single cycle and in each case the successor rule simplifies to $f(\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n) = \mathtt{a}_1$. Suppose $t > 1$. Let $\alpha_j = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ denote an arbitrary leaf of $\mathcal{T}$ with change index c_j. Let $\beta_1 = \mathtt{a}_1\cdots\mathtt{a}_{c_j-1}$, $\mathtt{y} = \mathtt{a}_{c_j}$, and $\beta_2 = \mathtt{a}_{c_j+1}\cdots\mathtt{a}_n$. Then $\alpha_j = \beta_1\mathtt{y}\beta_2$ and its parent is $\beta_1\mathtt{y}\ \beta_2$ for some $\mathtt{y}\ \in \Sigma$; the corresponding nodes in T are joined via the conjugate pair $(\mathtt{y}\beta_1\beta_2, \mathtt{y}\ \beta_1\beta_2)$. If $\mathcal{T} = \mathcal{T}_1$, let $\mathtt{x} = \mathtt{y}$; if $\mathcal{T} = \mathcal{T}_2$, let $\mathtt{x} = \text{first}(\mathtt{y}\ \beta_1\beta_2)$ with respect to T (recalling the definition of first in Section 2.2.1). Let $\mathcal{T}$ denote the concatenation tree obtained by removing α_j from $\mathcal{T}$. Similarly, let T denote the cycle-joining tree T with the leaf corresponding to α_j removed. Let $U_1 = \text{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-1})$ denote a rotation of $\text{RCL}(\mathcal{T}\ )$. By induction, U_1 is a universal cycle for $\mathbf{S}\ = \mathbf{S}_\mathsf{T} - [\alpha_j]$. Let $U_2 = \text{ap}(\alpha_j)$; it is a universal cycle for $[\alpha_j]$. Note that U_1 contains $\mathtt{x}\beta_2\beta_1$ and U_2 contains $\mathtt{y}\beta_2\beta_1$. The following claim will be proved later in Section 4.2.1.

B Claim 4. U_1 (considered cyclically) has prefix β_1 and suffix $\mathtt{x}\beta_2$.

Let $U_1 = \cdots\mathtt{x}\beta_2\beta_1$ and let $U_2 = \cdots\mathtt{y}\beta_2\beta_1$ be rotations of U_1 and U_2, respectively. Then by Theorem 1 and Claim 4, U_1 and U_2 can be joined via the conjugate pair $(\mathtt{x}\beta_2\beta_1, \mathtt{y}\beta_2\beta_1)$ to produce universal cycle U_1U_2, which is a rotation of U_1U_2, for $\mathbf{S}_\mathsf{T}$. Since U_1U_2 is a rotation of $U = \text{RCL}(\mathcal{T})$, the latter is also a universal cycle for $\mathbf{S}_\mathsf{T}$.

Clearly $\uparrow f_1 = \downarrow f_1$ with respect to the single PCR cycle $[\alpha_j]$; both functions are successor rules for U_2. Suppose $\mathcal{T} = \mathcal{T}_1$. From the induction hypothesis, $\uparrow f_1$ (with respect to T) is a successor rule for U_1. Since the two cycles U_1 and U_2 were joined via the conjugate pair $(\mathtt{x}\beta_2\beta_1, \mathtt{y}\beta_2\beta_1)$ to obtain U; the successors of only these two strings are altered. By the joining, the successor of $\mathtt{y}\beta_2\beta_1$ becomes the successor of $\mathtt{x}\beta_2\beta_1$ in U_1 which is precisely $\uparrow f_1(\mathtt{y}\beta_2\beta_1)$ with respect to T. The successor of $\mathtt{x}\beta_2\beta_1$ is $\mathtt{y}$, which is the same as $\uparrow f_1(\mathtt{x}\beta_2\beta_1)$ with respect to T. Thus, $\uparrow f_1$ (with respect to T) is a successor rule for U. A similar argument applies for $\mathcal{T} = \mathcal{T}_2$. J

I Remark 5. Consider a cycle-joining tree T where all chains in T have length $m = 2$. Then T induces a unique universal cycle with successor rule $\uparrow f_1 = \downarrow f_1$. Furthermore, if $k = 2$, $f_0 = \uparrow f_1 = \downarrow f_1$.

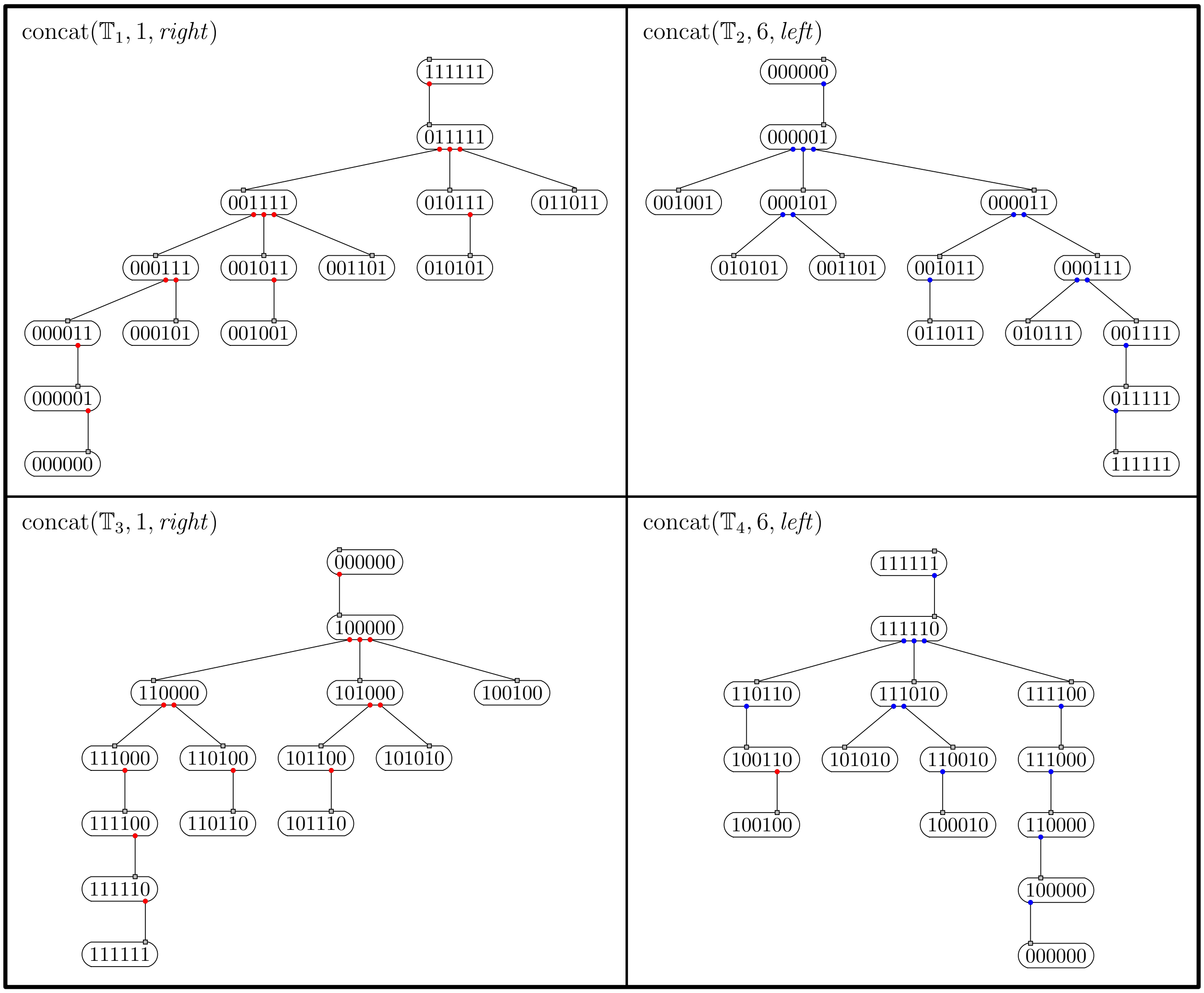


Figure 8 Concatenation trees for $n = 6$ based on $\mathtt{T}_1$, $\mathtt{T}_2$, $\mathtt{T}_3$, $\mathtt{T}_4$. These bifurcated ordered trees (BOTs) provide additional structure to the unordered cycle-joining trees from Figure 3. This structure provides the missing information for fully understanding the corresponding concatenation constructions. The gray box above each node indicates its change index.

4.1 Algorithmic details and analysis

A concatenation tree can be traversed to produce a universal cycle in $O(1)$-amortized time per symbol; but, it requires exponential space to store the tree. However, if the children of a given node α can be computed without knowledge of the entire tree, then we can apply Algorithm 1 to traverse a concatenation tree $\mathcal{T}$ in a space-efficient manner. The initial call is RCL(α,c, ℓ) where $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ is the root node with change index c. The variable ℓ is set to 1 for left concatenation trees; ℓ is set to 0 for right concatenation trees. The crux of the algorithm is the function CHILD(α, i) which returns $\mathtt{x}$ if there exists $\mathtt{x} \in \Sigma$ such that $\mathtt{a}_1\cdots\mathtt{a}_{i-1}\mathtt{x}\mathtt{a}_{i+1}\cdots\mathtt{a}_n$ is a child of α, or -1 otherwise. Since the underlying cycle-joining tree satisfies the Chain Property, if such an $\mathtt{x}$ exists then it is unique. In practice, the function must consider the acceptable range of α.

Let H denote the height of $\mathcal{T}$. Provided each call to CHILD(α, i) uses at most $O(n)$ space, the overall algorithm will require $O(n + H)$ space assuming α is passed by reference (or stored globally) and restored appropriately after each recursive call. The running time of Algorithm 1 depends on how efficiently the function CHILD(α, i) can be computed for each index i.

Theorem 6. *Let $\mathcal{T}$ be a concatenation rooted at α with change index c. The sequence resulting from a call to* RCL(α, c, ℓ) *is generated in $O(1)$-amortized time per symbol if (i) at each recursive step the work required by all calls to* CHILD(α, i) *is $O((t + 1)n)$, where t is the number of α's children, and (ii) the number of nodes in $\mathcal{T}$ that are periodic is less than some constant times the number of nodes that are aperiodic.*

Algorithm 1 Traversing a concatenation tree T in RCL order rooted at α with change index c

```
procedure RCL(α = a_1 ··· a_n, c, ℓ)
    for i ← c + ℓ to n do        ▷ Visit right-children
        x ← CHILD(α, i)
        if x = −1 then  RCL(a_1 ··· a_{i−1} x a_{i+1} ··· a_n, i, ℓ)
    p ← period of α
    PRINT(a_1 ··· a_p)
    for i ← 1 to c − 1 + ℓ do    ▷ Visit left-children
        x ← CHILD(α, i)
        if x = −1 then  RCL(a_1 ··· a_{i−1} x a_{i+1} ··· a_n, i, ℓ)
```

Proof. The work done at each recursive step is $O(n)$ plus the cost associated to all calls to CHILD(α, i). If condition (i) is satisfied, then the work can be amortized over the t children if $t \geq 1$, or onto the node itself if there are no children. Thus, each recursive node is the result of $O(n)$ work. By condition (ii), the total number of symbols output will be proportional to n times the number of nodes. Thus, each symbol is output in $O(1)$-amortized time. J

4.2 Properties of concatenation trees

Let $\alpha_1, \alpha_2, \dots, \alpha_t$ be the nodes of a concatenation tree $\mathcal{T}$ as they are visited in RCL order. Our proof of Claim 4 relies on properties exhibited between successive nodes in an RCL traversal of $\mathcal{T}$. The operations of the indices are taken modulo t, i.e., $\alpha_0 = \alpha_t$ and $\alpha_{t+1} = \alpha_1$. Recall that c_i denotes the change index of α_i. For the rest of this section, consider a node $\alpha_j = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_n$, for some $1 \leq j \leq t$, with change index c_j. Let $\beta_1 = \mathtt{a}_1\mathtt{a}_2 \cdots \mathtt{a}_{c_j-1}$, $\mathtt{y} = \mathtt{a}_{c_j}$ and let $\beta_2 = \mathtt{a}_{c_j+1} \cdots \mathtt{a}_n$; $\alpha_j = \beta_1 \mathtt{y} \beta_2$.

I **Lemma 7.** *If α_j is not an ancestor of α_{j+1}, then α_{j+1} has prefix β_1.*

Proof. Let $x = \alpha_j$ and $y = \alpha_{j+1}$. Following the notation from Figure 6, consider the four possible cases (b)(c)(e)(f) from Remark 2. If $t = 1$, the results are immediate. Suppose $t > 1$.

(b) r_m clearly has prefix β_1. Since the change index of r_m is less than or equal to c_j, and r_m only differs from its parent y at its change index, y must also have the prefix β_1.
(c) ℓ_i clearly has prefix β_1. The change index of ℓ_i is strictly less than the change index of ℓ_{i+1} and the two nodes differ only at those two indices. Thus, β_1 is a prefix of ℓ_{i+1}. Since y can only differ from ℓ_{i+1} in indices between the change index of ℓ_{i+1} and c_{j+1}, it must also have the prefix β_1.
(e) Trivial.
(f) All the nodes on the path from x up to the root and down to y must have change index greater than or equal to c_j. Thus each node, including y will have prefix β_1.

J

I **Lemma 8.** *If α_j is not an ancestor of α_{j+1}, and α_{j+1} has period $p < n$ with acceptable range $kp+1, \dots, kp+p$, then $c_j \leq kp + p$.*

Proof. Let $x = \alpha_j$ and $y = \alpha_{j+1}$. Following the notation from Figure 6, consider the four possible cases (b)(c)(e)(f) from Remark 2. If $t = 1$, the results are immediate. Suppose $t > 1$.

(b) The change index for r_m must be less than or equal to $kp + p$, and because α_j is a left descendant of r_m, c_j must be less than or equal to the change index of r_m. Thus, $c_j \leq kp + p$.
(c) c_j is less than or equal to the change index of ℓ_i, which is less than the change index of ℓ_{i+1}, which is less than or equal to c_{j+1}. Thus, $c_j < c_{j+1} \leq kp + p$.
(e) α_j is a left-descendant of α_{j+1} so clearly $c_j < c_{j+1} \leq kp + p$.
(f) c_j is less than or equal to the change index of the root, which is less than or equal to c_{j+1}. Thus, $c_j < c_{j+1} \leq kp + p$.

J

If α_j is not an ancestor of α_{j-1}, then from Remark 2, α_j is not the root node and thus has a parent $\beta_1\mathsf{y}\,\beta_2$ for some $\mathsf{y} \in \Sigma$. Recalling the definition of first in Section 2.2.1 with respect to the underlying cycle-joining tree T, let $\mathsf{x} = \text{first}(\mathsf{y}\,\beta_1\beta_2)$.

Lemma 9. *If α_j is not an ancestor of α_{j-1}, then*

1. *if $\mathcal{T}$ is a left concatenation tree then α_{j-1} has suffix $\mathsf{y}\,\beta_2$, and*
2. *if $\mathcal{T}$ is a right concatenation tree then α_{j-1} has suffix $\mathsf{x}\beta_2$.*

Proof. Let $x = \alpha_{j-1}$ and $y = \alpha_j$. Following notation from Figure 6, consider the four possible cases (a)(c)(d)(f) from Remark 2. If $t = 1$, the results are immediate. Suppose $t > 1$. Suppose $\mathcal{T}$ is a *left* concatenation tree.

(a) If $\ell_1 = y$, the result is immediate. Suppose $\ell_1 \neq y$. From the definition of y, ℓ_1 has suffix $\mathsf{y}\,\beta_2$ and change index strictly less than c_j. Since ℓ_1 differs from its parent x only at its change index, x must also have suffix $\mathsf{y}\,\beta_2$.

(c) If $\ell_{i+1} = y$, then it is already established that its parent a has suffix $\mathsf{y}\,\beta_2$. Otherwise, ℓ_{i+1} has suffix $\mathsf{y}\,\beta_2$ and change index less than c_j, which means that a again has suffix $\mathsf{y}\,\beta_2$. Since the change index of ℓ_i is less than the change index of ℓ_{i+1}, clearly x also has suffix $\mathsf{y}\,\beta_2$.

(d) Follows since $\mathcal{T}$ is a left concatenation tree.

(f) Let α_r be the root of $\mathcal{T}$. Clearly, α_r has suffix $\mathsf{y}\,\beta_2$ and $c_r < c_j$. Thus, x also will have suffix $\mathsf{y}\,\beta_2$.

Suppose $\mathcal{T}$ is a *right* concatenation tree. Let $\mathsf{x} = \text{first}(\mathsf{y}\,\beta_1\beta_2)$. This implies that all nodes on the path from $\beta_1\mathsf{x}\beta_2$ to $y = \alpha_j$ have change index c_j and the change index of $\beta_1\mathsf{x}\beta_2$ is not equal to c_j.

(a) If $\ell_1 = y$, then the change index of ℓ_1 is strictly less than the change index of x and the result follows as $\mathsf{x} = \mathsf{y}$. Suppose $\ell_1 \neq y$. If the change index of ℓ_1 is strictly less than c_j, then by the definition of x, ℓ_1 has suffix $\mathsf{x}\beta_2$. Thus, clearly x also has suffix $\mathsf{x}\beta_2$. Otherwise, the change index of ℓ_1 must be equal to c_j, and since it is a left-child of x, the change index of x is not equal to c_j. Thus, by the definition of x, x will be precisely $\beta_1\mathsf{x}\beta_2$.

(c) Recall this covers two cases where the children of a can be either be both left-children or both right-children. In either case, the change index of a can not be the same as the change index for $\ell i + 1$. Thus, following the same argument from (a), the node a will have suffix $\mathsf{x}\beta_2$. Since the change index of ℓ_i is less than the change index of ℓ_{i+1}, clearly x also has suffix $\mathsf{x}\beta_2$.

(d) Follows since $\mathcal{T}$ is a right concatenation tree.

(f) Let α_r be the root of $\mathcal{T}$. Clearly, α_r has suffix $\mathsf{x}\beta_2$ and $c_r \leq c_j$. Since all left descendants of the root will have change index strictly less than c_r, it follows that x also will have suffix $\mathsf{x}\beta_2$.

∎

Lemma 10. *If α_j is not an ancestor of α_{j-1}, and α_{j-1} has period $p < n$ with acceptable range $kp + 1, \ldots, kp + p$, then $c_j > kp$.*

Proof. Let $x = \alpha_{j-1}$ and $y = \alpha_j$. Following notation from Figure 6, consider the four possible cases (a)(c)(d)(f) from Remark 2. If $t = 1$, the results are immediate. Suppose $t > 1$.

(a) By the acceptable range, the change index for ℓ_1 must be greater than kp. Because α_j is a right descendant of ℓ_1, c_j must be greater than or equal to the change index of ℓ_1. Thus, $c_j > kp$.

(c) c_{j-1} is less than or equal to the change index of ℓ_i, which is less than the change index of ℓ_{i+1}, which is less than or equal to c_j. Thus, $kp < c_{j-1} < c_j$.

(d) α_j is a right-descendant of α_{j-1} so clearly $kp < c_{j-1} < c_j$.

(f) c_{j-1} is less than or equal to the change index of the root, which is less than c_j. Thus, $kp < c_{j-1} < c_j$.

∎

Lemma 11. *If α_j is periodic with period p and acceptable range $kp + 1, \ldots, kp + p$, then $\text{ap}(\alpha_j)^k$ is a prefix of α_{j+1}.*

Proof. If α_j is not an ancestor of α_{j+1}, the inequality $kp < c_j$ and Lemma 7 together imply $\mathrm{ap}(\alpha_j)^k$ is a prefix of α_{j+1}. It remains to consider cases (a) and (d) from Remark 2 where $x = \alpha_j$ is an ancestor of $y = \alpha_{j+1}$. For case (a), y is the leftmost right-descendent of x's first left-child ℓ_1. Since x is periodic, the change index of ℓ_1 is in α_j's acceptable range; it is greater than kp. y is a right descendant of ℓ_1 and thus $c_{j+1} > kp$, which means y differs from ℓ_1 only in indices greater than kp. For (d) clearly y differs only in indices greater than or equal to c_j, which means $c_{j+1} > kp$. Thus, for each case, $\mathrm{ap}(\alpha_j)^k$ is a prefix of α_{j+1}. J

Lemma 12. *If α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$, then $\mathrm{ap}(\alpha_j)^{k+1}$ is a prefix of $\mathrm{ap}(\alpha_j, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-1})$, which is a rotation of $\mathrm{RCL}(\mathcal{T})$, considered cyclically.*

Proof. Note that $|\,\mathrm{ap}(\alpha_j)^{k+1}| \leq n$. The proof is by induction on the number of nodes t. If $t = 1$, the result is trivial. Suppose the claim holds for any tree with less than $t > 1$ nodes. Let $\mathcal{T}$ have t nodes and let α_j be a leaf node of $\mathcal{T}$. If there are no periodic nodes, we are done. Otherwise, we first consider α_j, then all other periodic nodes in $\mathcal{T}$.

Suppose α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$. From Lemma 11, $\mathrm{ap}(\alpha_j)^k$ is a prefix of α_{j+1}. If α_{j+1} is aperiodic, then we are done. Suppose, then, that α_{j+1} is periodic with period p and acceptable range $k\;p\;+1, \ldots, k\;p\;+p$. Let $\mathcal{T}$ be the tree resulting from $\mathcal{T}$ when α_j is removed. It follows from (i) that $kp < c_j \leq k\;p\;+p$, which implies $\mathrm{ap}(\alpha_j)^k$ is a prefix of $\mathrm{ap}(\alpha_{j+1})^{k'+1}$. Additionally, since $\mathcal{T}$ has less than t nodes and α_{j+1} is periodic, $\mathrm{ap}(\alpha_{j+1})^{k'+1}$ is a prefix of $\mathrm{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-1})$ by our inductive assumption. Therefore, $\mathrm{ap}(\alpha_j)^{k+1}$ is a prefix of $\mathrm{ap}(\alpha_j, \alpha_{j+1}, \ldots \alpha_t, \alpha_1, \ldots, \alpha_{j-1})$.

Now consider α_{j-1}. If it is aperiodic, then by induction, the claim clearly holds for all periodic nodes in $\mathcal{T}$. Thus, assume α_{j-1} is periodic. By showing that $\mathrm{ap}(\alpha_{j-1}, \alpha_j), \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-2})$ has the desired prefix, then repeating the same arguments will prove the claim holds for every other periodic node in $\mathcal{T}$. Let α_{j-1} have period p and acceptable range $k\;\;p\;\;+1, \ldots, k\;\;p\;\;+p$. If α_j is aperiodic, Lemma 11 implies that $\mathrm{ap}(\alpha_{j-1})^{k''}$ is a prefix of $\alpha_j = \mathrm{ap}(\alpha_j)$ and thus the claim holds for α_{j-1}. If α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$, we already demonstrated that $\mathrm{ap}(\alpha_j)^{k+1}$ is a prefix of $\mathrm{ap}(\alpha_j, \ldots, \alpha_t, \alpha_1, \ldots, alpha_{j-1})$. From Lemma 11, $\mathrm{ap}(\alpha_{j-1})^{k''}$ is a prefix of α_j. Note that (i) and its proof handles cases (b)(c)(e)(f) from Remark 2 implying that $c_{j-1} < kp+p$ for these cases. Since α_{j-1} is not necessarily a leaf, we must also consider (a) and (d). In both cases, clearly $k\;\;p\;\; < c_j$. Either way, $k\;\;p\;\; < kp+p$, which means $\mathrm{ap}(\alpha_{j-1})^{k''}$ is a prefix of $\mathrm{ap}(\alpha_j)^{k+1}$. Thus, $\mathrm{ap}(\alpha_{j-1})^{k''+1}$ is a prefix of $\mathrm{ap}(\alpha_{j-1}, \alpha_j, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-2})$. J

Lemma 13. *If α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$, then $\mathrm{ap}(\alpha_j)^{n/p-k-1}$ is a suffix of α_{j-1}.*

Proof. If α_j is not an ancestor of α_{j-1}, the inequality $c_j \leq kp+p$ and Lemma 9 together imply $\mathrm{ap}(\alpha_j)^{n/p-k-1}$ is a suffix of α_{j-1}. It remains to consider cases (b) and (e) from Remark 2 where $y = \alpha_j$ is an ancestor of $x = \alpha_{j-1}$. For case (b), x is the rightmost left-descendent of y's last right-child r_m. Since y is periodic, the change index of r_m is in α_j's acceptable range; it is less than or equal to $kp+p$. x is a left descendant of r_m and thus $c_{j-1} \leq kp+p$, which means x differs from r_m only in indices less than or equal to $kp+p$. For (e) clearly x differs only in indices less than or equal to c_j, which means $c_{j-1} \leq kp+p$. Thus, for each case, $\mathrm{ap}(\alpha_j)^{n/p-k-1}$ is a suffix of α_{j-1}. J

Lemma 14. *If α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$, then $\mathrm{ap}(\alpha_j)^{n/p-k}$ is a suffix of $\mathrm{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_j)$, which is a rotation of $\mathrm{RCL}(\mathcal{T})$, considered cyclically.*

Proof. Let $q = n/p$. Note that $|ap(\alpha_j)^{q-k}| \leq n$. The proof is by induction on t. If $t = 1$, the result is trivial. Suppose the claim holds for any tree with less than $t > 1$ nodes. Let $\mathcal{T}$ have t nodes and let α_j be a leaf node of $\mathcal{T}$. If there are no periodic nodes, we are done. Otherwise, we first consider α_j, then all other periodic nodes in $\mathcal{T}$.

Suppose α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$. From Lemma 13, $\mathrm{ap}(\alpha_j)^{q-k-1}$ is a suffix of α_{j-1}. If α_{j-1} is aperiodic, then we are done. Suppose, then, that α_{j-1} is periodic with period p and acceptable range $k\;p\;+1, \ldots, k\;p\;+p$. Let $\mathcal{T}$ be the tree resulting from $\mathcal{T}$ when α_j is removed. It follows from (i) that $k\;p\; < c_j \leq kp+p$, or $n - kp - p < n - k\;p$, which implies $\mathrm{ap}(\alpha_j)^{q-k-1}$ is a suffix of $\mathrm{ap}(\alpha_{j-1})^{q'-k'}$, where $q\; = n/p$. Additionally, since $\mathcal{T}$ has less than t nodes and α_{j-1} is periodic, $\mathrm{ap}(\alpha_{j-1})^{q'-k'}$ is a suffix of $\mathrm{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-1})$ by our inductive assumption. Therefore, $\mathrm{ap}(\alpha_j)^{q-k}$ is a suffix of $\mathrm{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-1}, \alpha_j)$.

Now consider α_{j+1}. If it is aperiodic, then by induction the claim clearly holds for all periodic nodes in $\mathcal{T}$. Thus, assume α_{j+1} is periodic. By showing $\mathrm{ap}(\alpha_{j+2}, \ldots, \alpha_t, \alpha_1, \ldots, alpha_j, \alpha_{j+1})$ has the desired suffix, then repeating the same arguments will prove the claim holds for every other periodic node in $\mathcal{T}$. Let α_{j+1} have period p and acceptable range $k\ \ p\ \ +1, \ldots, k\ \ p\ \ +p$. If α_j is aperiodic, Lemma 13 implies that $\mathrm{ap}(\alpha_{j+1})^{q''-k''-1}$ is a suffix of $\alpha_j = \mathrm{ap}(\alpha_j)$ and thus the claim holds for α_{j+1}. If α_j is periodic with period p and acceptable range $kp+1, \ldots, kp+p$, we already demonstrated that $\mathrm{ap}(\alpha_j)^{q-k}$ is a suffix of $\mathrm{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_j)$. From Lemma 13, $\mathrm{ap}(\alpha_{j+1})^{q''-k''-1}$ is a suffix of α_j. Note that (i) and its proof handles cases (a)(c)(d)(f) from Remark 2 implying that $c_{j+1} > kp$ for these cases. Since α_{j+1} is not necessarily a leaf, we must also consider (b) and (e). In both cases, clearly $c_j \le k\ \ p\ \ +p$. Either way, $kp < k\ \ p\ \ +p$, which means $\mathrm{ap}(\alpha_{j+1})^{q''-k''-1}$ is a suffix of $\mathrm{ap}(\alpha_j)^{q-p}$. Thus, $\mathrm{ap}(\alpha_{j+1})^{q''-k''}$ is a suffix of $\mathrm{ap}(\alpha_{j+2}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_j, \alpha_{j+1})$. J

4.2.1 Proof of Claim 4

Recall that $U_1 = \mathrm{ap}(\alpha_{j+1}, \ldots, \alpha_t, \alpha_1, \ldots, \alpha_{j-1})$ and $\alpha_j = \beta_1 \mathsf{y} \beta_2$. Recall that α_j is assumed to be a leaf with parent $\beta_1 \mathsf{y}\ \beta_2$. Also, if $\mathcal{T} = \mathcal{T}_1$, then $\mathsf{x} = \mathsf{y}$; if $\mathcal{T} = \mathcal{T}_2$, $\mathsf{x} = \mathrm{first}(\mathsf{y}\ \beta_1\beta_2)$. Thus, from Lemma 7, α_{j+1} has prefix β_1 and from Lemma 9, α_{j-1} has suffix $\mathsf{x}\beta_2$. If α_{j-1} and α_{j+1} are aperiodic, then U_1 has prefix β_1 and suffix $\mathsf{x}\beta_2$ as required. If $t = 2$, then we are also done since U_1 is considered cyclically. It remains to consider the cases where α_{j-1} or α_{j+1} is periodic and $t > 2$. These cases apply the "acceptable range".

Suppose α_{j+1} has period $p < n$ and acceptable range $kp+1, \ldots, kp+p$. Since α_j is a leaf, $c_j \le kp+p$ by Lemma 8. Thus, β_1 is a prefix of $\mathrm{ap}(\alpha_{j+1})^{k+1}$ since β_1 is a prefix of α_{j+1} from Lemma 7. From Lemma 12, $\mathrm{ap}(\alpha_{j+1})^{k+1}$ is a prefix of U_1. Thus, β_1 is a prefix of U_1.

Suppose α_{j-1} has period $p < n$ and acceptable range $kp+1, \ldots, kp+p$. Since α_j is a leaf, $c_j > kp$ by Lemma 10. Thus, $\mathsf{x}\beta_2$ is a suffix of $\mathrm{ap}(\alpha_{j-1})^{n/p-k}$ since $\mathsf{x}\beta_2$ is a suffix of α_{j-1} from Lemma 9. From Lemma 14, $\mathrm{ap}(\alpha_{j-1})^{n/p-k}$ is a suffix of U_1. Thus, $\mathsf{x}\beta_2$ is a suffix of U_1.

5 Applications

In this section we highlight how our concatenation tree framework can be applied to a variety of interesting objects including permutations, weak orders, orientable sequences, and DB sequences with related universal cycles. For each object, we define a PCR-based cycle-joining tree T that satisfies the Chain Property, where each node is a necklace (the lex smallest representative). Then we apply the concatenation tree framework and Algorithm 1 to construct the corresponding universal cycles in $O(1)$-amortized time per symbol.

5.1 De Bruijn sequences and relatives

Recall that pcr_1, pcr_2, pcr_3, and pcr_4 are stated generally for subtrees of the corresponding cycle-joining trees $\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4$; they focus on binary strings, and thus satisfy the Chain Property. Though we focus on the binary case, these trees can be generalized to larger alphabets following the theory in [24]. For instance, the parent rule used to create T_1 can be generalized to "increment the last non-$(k-1)$" where the alphabet is $\Sigma = \{0, 1, \ldots, k-1\}$.

Theorem 15. *Let T_1, T_2, T_3, T_4 be subtrees of $\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3, \mathsf{T}_4$, respectively. For any $1 \le c \le n$ and $\ell \in \{left, right\}$:*

- $U_1 = \mathrm{RCL}(\mathrm{concat}(T_1, c, \ell))$ *is a universal cycle for* $\mathbf{S}_{T_1}$ *with successor rule* pcr_1.
- $U_2 = \mathrm{RCL}(\mathrm{concat}(T_2, c, \ell))$ *is a universal cycle for* $\mathbf{S}_{T_2}$ *with successor rule* pcr_2.
- $U_3 = \mathrm{RCL}(\mathrm{concat}(T_3, c, \ell))$ *is a universal cycle for* $\mathbf{S}_{T_3}$ *with successor rule* pcr_3.
- $U_4 = \mathrm{RCL}(\mathrm{concat}(T_4, c, \ell))$ *is a universal cycle for* $\mathbf{S}_{T_4}$ *with successor rule* pcr_4.

Proof. The results follow immediately from Remark 5 and Theorem 3. J

Interesting subtrees applied to the above theorem include nodes with: (i) a lower bound on weight (T_1 and T_4), (ii) an upper bound on weight (T_2 and T_3), (iii) forbidden 0^s substring (T_1 and T_4), (iv) forbidden 1^s substring (T_2 and T_3). Universal cycles for strings with bounded weight (based on T_1 and T_2) [40, 42, 43] and universal cycles with forbidden 0^s (based on T_1) [20, 44] have been previously studied; the latter has found recent application in quantum key distribution schemes [7]. Theorem 15 generalizes these result and provides a connection between the concatenation constructions and corresponding successor rules.

If the subtrees in Theorem 15 are T_1, T_2, T_3, and T_4, respectively, the universal cycles are DB sequences. Specifically, let

- $\mathcal{T}_1 = \text{concat}(\mathsf{T}_1, 1, \textit{right})$,
- $\mathcal{T}_2 = \text{concat}(\mathsf{T}_2, n, \textit{left})$,
- $\mathcal{T}_3 = \text{concat}(\mathsf{T}_3, 1, \textit{right})$, and
- $\mathcal{T}_4 = \text{concat}(\mathsf{T}_4, n, \textit{left})$.

Recall that the Granddaddy DB sequence can be constructed by concatenating the aperiodic prefixes of necklaces as they appear in lexicographic order; it is known to have the successor rule pcr_1, and the sequence can be generated in $O(1)$-amortized time per bit.

Corollary 16. $\text{RCL}(\mathcal{T}_1)$ *is the Granddaddy DB sequence with successor rule* pcr_1.

Proof. $\mathcal{T}_1$ is based on the "last 0" cycle-joining tree rooted at 1^n, the change index of the root is 1, and the tree is right-concatenation tree. Thus, the representatives of each node are necklaces (flipping the last 0 of a necklace yields a necklace), where the change index of each child is after the last 0. This means each child is a right-child that is lexicographically smaller than its parent, and the children are ordered lexicographically left to right. Therefore, RCL order is a post-order traversal of $\mathcal{T}_1$ that visits the necklaces $\mathbf{N}_2(n)$ in lexicographic order. Thus, $\text{RCL}(\mathcal{T}_1)$ is the Granddaddy DB sequence, and by Theorem 15 it has successor rule pcr_1. ◀

The Grandmama DB sequence can be constructed by concatenating the aperiodic prefixes of necklaces as they appear in co-lexicographic order; it is known to have the successor rule pcr_2, and the sequence can be generated in $O(1)$ amortized time per bit.

Corollary 17. $\text{RCL}(\mathcal{T}_2)$ *is the Grandmama DB sequence with successor rule* pcr_2.

Proof. $\mathcal{T}_2$ is based on the "first 1" cycle-joining tree rooted at 0^n, the change index of the root is n, and the tree is left-concatenation tree. Thus, the representatives of each node are necklaces (flipping the first 1 of a necklace yields a necklace), where the change index of each child is before the first 1. This means each child is a left-child that is lexicographically larger than its parent and the children are ordered co-lexicographically. Therefore, RCL order is a pre-order traversal of $\mathcal{T}_2$ that visits the necklaces $\mathbf{N}_2(n)$ in co-lexicographic order. Thus, $\text{RCL}(\mathcal{T}_2)$ is the Grandmama DB sequence, and by Theorem 15 it has successor rule pcr_2. ◀

Though the concatenation-tree framework is not necessary to obtain a more efficient DB sequence construction for the Granddaddy and Grandmama, there is a significant improvement in the simplicity of verifying both the correctness and the equivalence of the concatenation and successor rule constructions.

We now answer an unproved claim about the correspondence between the DB sequence generated by pcr_3 and a concatenation construction from [20]. In particular, let the representative of each necklace class be the necklace with the initial prefix of 0s rotated to the suffix, so each representative (except 0^n) begins with 1. The construction from [20] concatenates the aperiodic prefixes of these representatives as they appear in reverse lexicographic order; here we name it the *Granny* DB sequence. As an example, see the sequence generated by pcr_3 in Table 1. This sequence can also be generated in $O(1)$-amortized time per bit [45].

Corollary 18. $\text{RCL}(\mathcal{T}_3)$ *is the Granny DB sequence with successor rule* pcr_3.

Proof. $\mathcal{T}_3$ is based on the "last 1" cycle-joining tree rooted at 0^n, the change index of the root is 1, and the tree is right-concatenation tree. Thus, it is not difficult to observe recursively that the representative of each non-root node is a necklace with the initial prefix of 0s is rotated to the suffix, so it begins with a 1. Furthermore, the change index of each child is in the rotated suffix of 0s. This means each child is a right-child that is lexicographically smaller than its parent and the children are ordered in reverse lexicographic order. Therefore, RCL order is a post-order traversal of $\mathcal{T}_3$ that visits the representatives in reverse-lexicographic order. Thus, $\text{RCL}(\mathcal{T}_3)$ is the Granny DB sequence, and by Theorem 15 it has successor rule pcr_3. ◀

Recall that the question of whether or not there existed a "simple" concatenation construction for pcr_4 was the motivating question that lead to the discovery of concatenation trees. Unfortunately, it appears as the though the resulting RCL traversal is not so simple; each node representative appears to require recursive information about its parent (and hence the tree structure). Here, we name the sequence constructed by $\text{RCL}(\mathcal{T}_4)$ the *Grandpa* DB sequence. Experimental evidence indicates an algorithm that runs in $O(1)$-amortized time per bit may exist using the concatenation tree framework; however, the analysis appears non-trivial.

Corollary 19. $\mathrm{RCL}(\mathcal{T}_4)$ *is the Grandpa DB sequence with successor rule* pcr_4.

5.2 Universal cycles for shorthand permutations

A *shorthand* permutation is a length $n{-}1$ prefix of some permutation $\mathsf{p}_1\mathsf{p}_2\cdots\mathsf{p}_n$. Let $\mathbf{SP}(n)$ denote the set of shorthand permutations of order n. For example, $\mathbf{SP}(3) = \{12, 13, 21, 23, 31, 32\}$. Note that $\mathbf{SP}(n)$ is closed under rotation. The necklace classes of $\mathbf{SP}(n)$ can be joined into a PCR-based cycle-joining tree via the following parent rule [24], where each cycle representative is a necklace.

> **Parent rule for cycle-joining the necklaces in $\mathbf{SP}(n)$:** Let r denote the root $12\cdots(n-1)$. Let σ denote a non-root node where z is the missing symbol. If $\mathsf{z} = n$, let $j > 1$ denote the smallest index such that $\mathsf{p}_j < \mathsf{p}_{j-1}$, otherwise let j denote the index of $\mathsf{z}+1$. Then
>
> $$\mathrm{par}(\sigma) = \mathsf{p}_1\cdots\mathsf{p}_{j-1}\mathsf{z}\mathsf{p}_{j+1}\cdots\mathsf{p}_{n-1}.$$

Let T_{perm} be the cycle-joining tree derived from the above parent rule; it satisfies the Chain Property. Observe that a node $\sigma = \mathsf{p}_1\mathsf{p}_2\cdots\mathsf{p}_{n-1}$ in T will have at most two children. In particular, if z is the missing symbol, $\mathsf{p}_1\cdots\mathsf{p}_j\mathsf{z}\mathsf{p}_{j+1}\cdots\mathsf{p}_{n-1}$ is a child of σ if and only if (i) $\mathsf{p}_j = \mathsf{z}{-}1$, or (ii) $\mathsf{p}_j = n$, $\mathsf{p}_1\cdots\mathsf{p}_{j-1}$ is increasing, and $\mathsf{z} < \mathsf{p}_{j-1}$. Figure 9 illustrates T_{perm} and $\mathcal{T}_{perm} = \mathrm{concat}(\mathsf{T}_{perm}, n{-}1, \mathit{left})$ for $n = 4$. Let $U_{perm} = \mathrm{RCL}(\mathcal{T}_{perm})$, when n is understood. Then, for $n = 4$:

$$U_{perm} = 123\ 124\ 132\ 143\ 243\ 142\ 134\ 234.$$

A unique successor rule f_{perm} for the universal cycle derived from T_{perm} can be computed in $O(n)$ time [24].

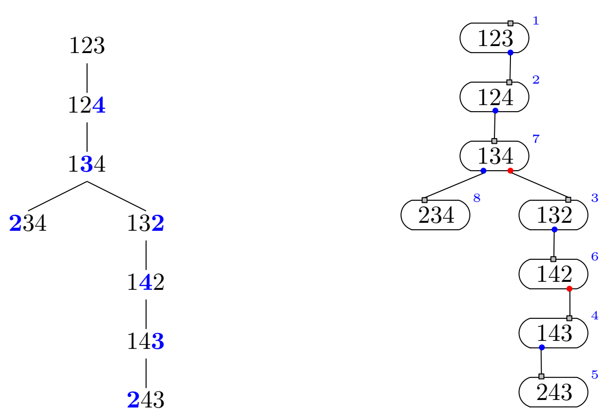


Figure 9 A cycle joining tree T_{perm} of shorthand permutation necklaces for $n = 4$, and the corresponding concatenation tree $\mathcal{T}_{perm}$ illustrating the RCL order.

Theorem 20. *U_{perm} is a universal cycle for $\mathbf{SP}(n)$ with successor rule f_{perm}. Moreover, U_{perm} can be constructed in $O(1)$-amortized time per symbol using $O(n^2)$ space.*

Proof. Since each chain in T_{perm} has length at most 2, f_{perm} is unique and $f_{perm} = \uparrow f_1 = \downarrow f_1$ (see Remark 5). Thus, by Theorem 3, U_{perm} is a universal cycle for $\mathbf{SP}(n)$ with successor rule f_{perm}. As noted earlier, each node σ in T_{perm} has at most two nodes; they can easily be determined in linear time with a single scan of σ and the values can be stored using a constant amount of space. Thus, by Theorem 6, U_{perm} can be constructed in $O(1)$-amortized time per symbol. The space required by the algorithm is proportional to the height of T_{perm}; each recursive call requires a constant amount of space. Consider a node $\sigma = \mathsf{p}_1\mathsf{p}_2\cdots\mathsf{p}_{n-1}$ in T_{perm}, where j is the smallest index such that $\mathsf{p}_j < \mathsf{p}_{j-1}$. By applying at most n applications of the parent rule, σ has an ancestor whose length-j prefix is increasing and the j-th symbol is n. Thus, after at most n^2 applications of the parent rule, σ has an ancestor σ that is increasing and ends with n. After at most n applications of the parent rule, σ will have the root r as an ancestor. Thus, the height of T_{perm} is $O(n^2)$. ∎

Efficient concatenation constructions of universal cycles for shorthand permutations are known [27, 38]; however (i) there is no clear connection between their construction and corresponding successor rule and (ii) they do not have underlying PCR-based cycle-joining trees.

5.3 Universal cycles for weak orders

Recall that $\mathbf{W}(n)$ denotes the set of weak orders of order n; it is closed under rotation. Weak orders of order n are counted by the ordered Bell or Fubini numbers (OEIS A000670 [2]). The first construction of a universal cycle for $\mathbf{W}(n)$ defined the upcoming PCR-based cycle-joining tree, where the cycle-representatives (nodes) are the lex-smallest representatives (necklaces) [46]. Let $\omega = \mathtt{w}_1\mathtt{w}_2\cdots\mathtt{w}_n$ denote a string in $\mathbf{W}(n)$. Let $n_\omega(i)$ denote the number of occurrences of the symbol i in ω. Let $\mathbf{W_1}(n)$ denote the set of all weak orders of order n with no repeating symbols other than perhaps 1.

> **Parent rule for cycle-joining the necklaces in $\mathbf{W}(n)$:** Let r denote the root 1^n. Let $\omega = \mathtt{w}_1\mathtt{w}_2\cdots\mathtt{w}_n$ denote a non-root node. If ω $\mathbf{W_1}(n)$, let j denote the index of the symbol $n_\omega(1)+1$ and let $\mathtt{x} = 1$; otherwise let j be the largest index containing a repeated (non-1) symbol and let $\mathtt{x} = \mathtt{w}_j + n_\omega(\mathtt{w}_j) - 1$. Then
>
> $\text{par}(\omega) = \mathtt{w}_1\cdots\mathtt{w}_{j-1}\mathtt{x}\mathtt{w}_{j+1}\cdots\mathtt{w}_n.$

Let $\mathtt{T}_{weak}$ be the cycle-joining tree derived from the above parent rule; it clearly satisfies the Chain Property. Figure 2 illustrates both $\mathtt{T}_{weak}$ and $\mathcal{T}_{weak} = \text{concat}(\mathtt{T}, n, \textit{left})$ for $n = 4$. A successor rule f_{weak} for the universal cycle based on $\mathtt{T}_{weak}$ can be computed in $O(n)$ time [46].

Our goal is to apply Theorem 6 to construct an universal cycle for weak orders in $O(1)$-amortized time. Consider a node $\omega = \mathtt{w}_1\mathtt{w}_2\cdots\mathtt{w}_n$ in $\mathtt{T}_{weak}$. Let $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ denote a sequence such that $\mathtt{c}_i = \mathtt{x}$ if there exists an $\mathtt{x}$ such that the necklace of $\mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{x}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$ is a child of ω in $\mathtt{T}$, or -1 otherwise. Note that $\mathtt{x}$ is unique since $\mathtt{T}_{weak}$ satisfies the Chain Property.

| **Lemma 21.** *If $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n$ is a necklace then $\beta = \mathtt{a}_j\cdots\mathtt{a}_n\mathtt{a}_1\cdots\mathtt{a}_{i-1}\mathtt{x}\mathtt{a}_{i+1}\cdots\mathtt{a}_{j-1}$ is not a necklace for any $1 \le i < j \le n$ where $\mathtt{x} < \mathtt{a}_i$.*

Proof. Suppose $1 \le i < j \le n$. Since α is a necklace, $\mathtt{a}_j\cdots\mathtt{a}_n\mathtt{a}_1\cdots\mathtt{a}_i \ge \mathtt{a}_1\cdots\mathtt{a}_{n-j+i+1}$. If β is a necklace then the length i prefix of β must be less than or equal to $\mathtt{a}_1\cdots\mathtt{a}_{i-1}\mathtt{x}$ which is less than $\mathtt{a}_1\cdots\mathtt{a}_i$ since $\mathtt{x} < \mathtt{a}_i$. Contradiction. J

| **Lemma 22.** *Let $\omega = \mathtt{w}_1\mathtt{w}_2\cdots\mathtt{w}_n$ be a necklace in $\mathbf{W}(n) \setminus \mathbf{W}_1(n)$. Given $1 \le i \le n$, if $\mathtt{w}_i > 1$ let $\mathtt{y}$ be the largest symbol smaller than $\mathtt{w}_i$ in ω; otherwise, let $\mathtt{y} = 1$. If $\mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{y}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$ is not a necklace then $\mathtt{c}_i = -1$.*

Proof. From the parent rule, a 1 is never changed to $\mathtt{x}$. Thus, if $\mathtt{y} = 1$ then $\mathtt{c}_i = -1$. Suppose $\mathtt{y} > 1$. Let ω be the necklace of $\mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{y}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$. Since ω clearly begins with a 1, from Lemma 21, ω starts with $\mathtt{w}_\mathtt{j}$ for some $j < i$. Since ω and ω are both necklaces, $\mathtt{w}_1\cdots\mathtt{w}_{i-j} = \mathtt{w}_j\cdots\mathtt{w}_{i-1}$. Consider all occurrences of $\mathtt{x} \in \omega$. If $\mathtt{x}$ does not appear before index j, then since $\mathtt{w}_1\cdots\mathtt{w}_{i-j} = \mathtt{w}_j\cdots\mathtt{w}_{i-1}$, it must appear at an index after i. Either way, the $\mathtt{y}$ slotted into position i of ω is not the right most repeated symbol in the corresponding necklace representative ω, and thus the parent of ω is not ω. Thus, from the parent rule, $\mathtt{c}_i = -1$. J

The sequence $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ can be determined for a necklace ω by considering the following two cases from the parent rule.

1. Suppose $\omega \in \mathbf{W}_1$. For each $1 \le j \le n$, if $\mathtt{w}_j = 1$ then $\mathtt{c}_j = n_\omega(1)$; otherwise, $\mathtt{c}_j = -1$
2. Suppose $\omega \notin \mathbf{W}_1$. Let i denote the largest index such that $\mathtt{w}_i > 1$ and $n_\omega(\mathtt{w_i}) > 1$; all symbols in $\mathtt{w}_{i+1}\cdots\mathtt{w}_n$ are unique within ω. Consider $1 \le j \le n$. If $\mathtt{w}_j = 1$, then clearly by the parent rule $\mathtt{c}_i = -1$. Otherwise, let $\mathtt{x}$ denote the largest symbol in ω less than $\mathtt{w}_j$ and let $\omega = \mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{x}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$. If $1 \le j < i$, then $\mathtt{x}$ is not the rightmost (non-1) repeated symbol in ω, and by the parent rule $\mathtt{c}_j = -1$. If $i = j$, then by the parent rule, then $\mathtt{c}_i = -1$, since $\mathtt{w}_i$ is a repeated symbol. Suppose $i+1 \le j \le n$. If $\mathtt{x} = 1$ or ω is not a necklace, then by the parent rule and Lemma 22, respectively, $\mathtt{c}_j = -1$. Otherwise, suppose $\mathtt{x} > 1$ and ω is a necklace. Then $\mathtt{c}_j = \mathtt{x}$ if $\mathtt{x}$ does not appear in $\mathtt{w}_{j+1}\cdots\mathtt{w}_n$; otherwise, $\mathtt{x}$ is not the rightmost (non-1) repeated symbol in ω, and thus $\mathtt{c}_j = -1$.

Algorithm 2 Computing $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ for given node $\omega = \mathtt{w}_1\mathtt{w}_2\cdots\mathtt{w}_n$.

1: $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n \leftarrow (-1)^n$
2: **if** $\omega \in \mathbf{W}_1(n)$ **then** ▷ Case 1
3:     **for** i **from** 1 **to** n **do**
4:         **if** $\mathtt{w}_i = 1$ **then** $\mathtt{c}_i \leftarrow n_\omega(1)$
5: **if** $\omega \notin \mathbf{W}_1(n)$ **then** ▷ Case 2
6:     $v_1v_2\cdots v_n \leftarrow 0^n$ ▷ v_i is set to 1 if symbol i is visited in the following loop
7:     **for** i **from** n **down to** 1 **do**
8:         **if** $\mathtt{w}_i > 1$ **and** $n_\omega(\mathtt{w}_i) > 1$ **then break**
9:         **else**
10:             $\mathtt{x} \leftarrow$ the largest symbol in ω less than $\mathtt{w}_i$, or 0 if $\mathtt{w}_i = 1$
11:             **if** $\mathtt{x} > 1$ **and** IsNecklace($\mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{x}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$) **and** $v_\mathtt{x} = 0$ **then** $\mathtt{c}_i \leftarrow \mathtt{x}$
12:         $v_{\mathtt{w}_i} \leftarrow 1$

Applying the two cases above, Algorithm 2 computes the values $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ for ω.
If $n \leq 8$, there is at most one call to IsNecklace on line 11 of Algorithm 2 that returns false, for a given input ω. For $n = 9$, there are 10 strings for which the function returns false twice. One of these strings is $\omega = 147914816$, where the highlighted symbols correspond to the indices i where such a call returns false. The corresponding strings tested by the algorithm (in the order tested) are 147914814 and 147914616. Neither are necklaces. After the second test, the following lemma demonstrates that the next non-1 symbol $\mathtt{w}_i$ considered by the **for** loop (line 7) must be a repeated symbol in ω.

▶ **Lemma 23.** *There are at most two calls to* IsNecklace($\mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{x}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$) *in Algorithm 2 (line 11) where* $\mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{x}\mathtt{w}_{i+1}\cdots\mathtt{w}_n$ *is not a necklace.*

Proof. We trace Algorithm 2 and the **for** loop (line 7) noting that $\omega = \mathtt{w}_1\mathtt{w}_2\cdots\mathtt{w}_n$ is a weak order necklace representative not in $\mathbf{W}_1(n)$. We demonstrate that if IsNecklace returns false twice, then the next iteration of the loop where $\mathtt{w}_i > 1$ must have $n_\omega(\mathtt{w}_i) > 1$. Thus, the loop breaks (line 8) and there are no further calls to IsNecklace.

Consider two iterations of the **for** loop (line 7) where the iterator has value i and j, respectively, such that both iterations make a call to IsNecklace (line 11) that returns false. Furthermore, assume $j < i$ are the two largest values such that this is the case. Let $\alpha_i = \mathtt{w}_1\cdots\mathtt{w}_{i-1}\mathtt{x}_i\mathtt{w}_{i+1}\cdots\mathtt{w}_n$ noting that $\mathtt{w}_i > \mathtt{x}_i > 1$ (lines 10 and 11) and $n_\omega(\mathtt{w}_i) = 1$ (line 8). Since α_i is not a necklace, by Lemma 21, there exists some largest index $1 \leq t_i \leq i$ such that the rotation of α starting from index t_i is a necklace. Thus, $\mathtt{w}_{t_i}\cdots\mathtt{w}_{i-1} \leq \mathtt{w}_1\cdots\mathtt{w}_{i-t_i}$. Since ω is not a necklace, $\mathtt{w}_{t_i}\cdots\mathtt{w}_{i-1} \geq \mathtt{w}_1\cdots\mathtt{w}_{i-t_i}$. Thus, $\mathtt{w}_{t_i}\cdots\mathtt{w}_{i-1} = \mathtt{w}_1\cdots\mathtt{w}_{i-t_i}$ (*). Define α_j, $\mathtt{x}_j$, and t_j similarly, which means $\mathtt{w}_j > \mathtt{x}_j > 1$ and $n_\omega(\mathtt{w}_j) = 1$. If $t_i \leq j < i$, then $n_\omega(\mathtt{w}_j) > 1$ by (*), a contradiction. Thus, $t_j \leq j < t_i$. Since $n_\omega(\mathtt{w}_j) = 1$, $\mathtt{w}_j \neq \mathtt{w}_i$. ◀

Let $U_{weak} = \mathrm{RCL}(\mathcal{T}_{weak})$.

▶ **Theorem 24.** *U_{weak} is a universal cycle for $\mathbf{W}(n)$ with successor rule f_{weak}. Moreover, U_{weak} can be constructed in $O(1)$-amortized time per symbol using $O(n^2)$ space.*

Proof. Based on the parent rule for $\mathtt{T}_{weak}$, every chain in $\mathcal{T}_{weak}$ has length $m = 2$. From Remark 5, $f_{weak} = \uparrow f_1 = \downarrow f_1$, and Theorem 3 implies that U_{weak} is a universal cycle for $\mathbf{W}(n)$ with successor rule f_{weak}.

To generate U_{weak}, Algorithm 1 can apply Algorithm 2 to determine the children of a node by computing and storing $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$; each recursive call requires $O(n)$ space. The height of $\mathtt{T}_{weak}$ is $O(n)$; the path from any leaf to the root requires less than n applications of $\mathrm{par}(\omega)$ to break all the non-1 ties, and then less than n applications of $\mathrm{par}(\omega)$ to convert all the non-1s to 1s. Thus, the construction requires $O(n^2)$ space. The time to generate U_{weak} depends on how efficiently we can compute $\mathtt{c}_1\mathtt{c}_2\cdots\mathtt{c}_n$ in Algorithm 2. The values $n_\omega(\mathtt{w}_i)$ can be computed in $O(n)$ time cumulatively. Applying these values, the cumulative time to compute $\mathtt{x}$ at line 10 is also $O(n)$, since each $\mathtt{w}_i > 1$ must be unique by line 8. Thus, excluding the calls to IsNecklace, the algorithm runs in $O(n)$ time. Each call to IsNecklace (which requires $O(n)$ time [5]) that returns true leads to a child (some $c_i > -1$). From Lemma 23, there at most two calls to IsNecklace that return false. Thus, the total work by Algorithm 2 is $O((t+1)n)$, where t is the number of children of the input node ω. Hence by Theorem 6, U_{weak} can be constructed in $O(1)$-amortized time. ◀

5.4 Orientable sequences

An *orientable sequence* is a universal cycle for a set $\mathbf{S} \subseteq \{0,1\}^n$ such that if $\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n \in \mathbf{S}$, then its reverse $\mathtt{a}_n\cdots\mathtt{a}_2\mathtt{a}_1 \notin \mathbf{S}$. Thus, $\mathbf{S}$ does not contain palindromes. Orientable sequences do not exist for $n < 5$, and a maximal length orientable sequence for $n = 5$ is 001011. Somewhat surprisingly, the maximal length of binary orientable sequences are known only for $n = 5, 6, 7$. Orientable sequences were introduced in [11] with applications related to robotic position sensing. They established upper and lower bounds for their maximal length; the lower bound is based on the *existence* of a PCR-based cycle-joining tree, though no construction of such a tree was provided.

A *bracelet class* is an equivalence class of strings under rotation and reversal; its lexicographically smallest representative is a *bracelet*. A bracelet α is *symmetric* if $[\alpha] = [\alpha^R]$; otherwise it is *asymmetric*. Let $\mathbf{A}(n)$ denote the set of all binary asymmetric bracelets of length n. For example, $\mathbf{A}(8) = \{00001011, 00010011, 00010111, 00101011, 00101111, 00110111\}$. If $\mathbf{A}(n) = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$, let $\mathbf{O}(n) = [\alpha_1] \cup [\alpha_2] \cup \cdots \cup [\alpha_t]$.

Motivated by the work in [11], a cycle-joining tree T_{orient} for $\mathbf{A}(n)$ was discovered leading to the construction of an orientable sequence with asymptotically optimal length [21]. The parent rule combines three of the four "simple" parent rules defined earlier for PCR-based cycle joining trees; it applies the following functions where $\alpha = \mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_n \in \mathbf{A}(n)$:

- $\text{first1}(\alpha)$ be the necklace $\mathtt{a}_1\cdots\mathtt{a}_{i-1}\mathbf{0}\mathtt{a}_{i+1}\cdots\mathtt{a}_n$, where i is the index of the first 1 in α;
- $\text{last1}(\alpha)$ be the necklace in $[\mathtt{a}_1\mathtt{a}_2\cdots\mathtt{a}_{n-1}\mathbf{0}]$;
- $\text{last0}(\alpha)$ be the necklace $\mathtt{a}_1\cdots\mathtt{a}_{j-1}\mathbf{1}\mathtt{a}_{j+1}\cdots\mathtt{a}_n$, where j is the index of the last 0 in α.

Parent rule for cycle-joining $\mathbf{A}(n)$: Let r denote the root $0^{n-4}1011$. Let α denote a non-root node in $\mathbf{A}(n)$. Then

$$\text{par}(\alpha) = \text{ the first asymmetric bracelet in the list: } \ \text{first1}(\alpha),\ \text{last1}(\alpha),\ \text{last0}(\alpha).$$

Let T_{orient} be the cycle-joining tree derived from the above parent rule. Figure 10 illustrates T_{orient} together with $\mathcal{T}_{orient} = \text{concat}(\mathsf{T}_{orient}, n, \textit{right})$ for $n = 8$; an RCL traversal of $\mathcal{T}_{orient}$ produces the following orientable sequence of length 48:

00001011 11001011 10011011 10001001 10001011 00101011.

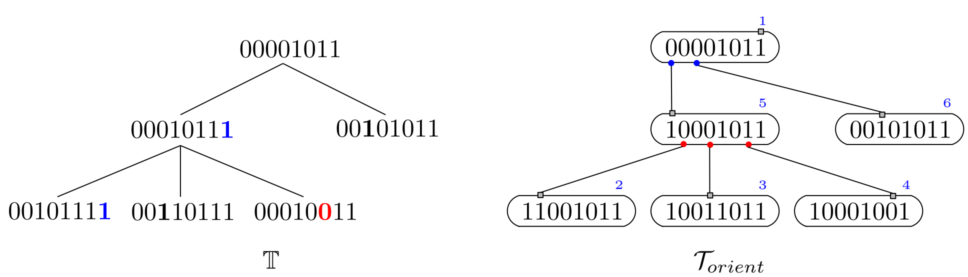


Figure 10 A cycle-joining tree for $\mathbf{A}(8)$ based on the parent rule $\text{par}(\alpha)$ followed by a corresponding right concatenation tree $\mathcal{T}_{orient}$ that illustrates the RCL order.

A successor rule f_{orient} based on T_{orient} constructs the corresponding orientable sequence in $O(n)$-time per symbol [21]. The following result is proved by applying Theorem 3 and Theorem 6, where $U_{orient} = \text{RCL}(\mathcal{T}_{orient})$.

Theorem 25 ([21]). *U_{orient} is a universal cycle for $\mathbf{O}(n)$ with successor rule f_{orient}. Moreover, U_{orient} can be constructed in $O(1)$-amortized time per symbol using $O(n^2)$ space.*

References

1 De Bruijn sequence and universal cycle constructions (2023). http://debruijnsequence.org.
2 The On-Line Encyclopedia of Integer Sequences, published electronically at https://oeis.org, 2010, sequences A006013, A000670.

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

4 BANKEVICH, A., BZIKADZE, A. V., KOLMOGOROV, M., ANTIPOV, D., AND PEVZNER, P. A. Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads. *Nature Biotechnology 40*, 7 (Jul 2022), 1075–1081.

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

6 BROCKMAN, G., KAY, B., AND SNIVELY, E. On universal cycles of labeled graphs. *Electronic Journal of Combinatorics 17* (200), R4.

7 CHEE, Y. M., ETZION, T., NGUYEN, T. L., TA, D. H., TRAN, V. D., AND VU, V. K. Maximum length RLL sequences in de Bruijn graph, https://arxiv.org/abs/2403.01454, 2024.

8 CHUNG, F., DIACONIS, P., AND GRAHAM, R. Universal cycles for combinatorial structures. *Discrete Mathematics 110*, 1-3 (1992), 43–59.

9 COMPEAU, P. E., PEVZNER, P. A., AND TESLER, G. How to apply de Bruijn graphs to genome assembly. *Nature biotechnology 29*, 11 (2011), 987–991.

10 CURTIS, D., HINES, T., HURLBERT, G., AND MOYER, T. Near-universal cycles for subsets exist. *SIAM Journal on Discrete Mathematics 23*, 3 (2009), 1441–1449.

11 DAI, Z.-D., MARTIN, K., ROBSHAW, B., AND WILD, P. Orientable sequences. In *Cryptography and Coding III (M.J.Ganley, ed.)* (1993), Oxford University Press, pp. 97–115.

12 DRAGON, P. B., HERNANDEZ, O. I., SAWADA, J., WILLIAMS, A., AND WONG, D. Constructing de Bruijn sequences with co-lexicographic order: The k-ary Grandmama sequence. *European Journal of Combinatorics 72* (2018), 1–11.

13 ELDERT, C., GRAY, H., GURK, H., AND RUBINOFF, M. Shifting counters. *AIEE Trans. 77* (1958), 70–74.

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

15 ETZION, T., AND LEMPEL, A. Construction of de Bruijn sequences of minimal complexity. *IEEE Transactions on Information Theory 30*, 5 (September 1984), 705–709.

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

17 FREDRICKSEN, H., AND KESSLER, I. Lexicographic compositions and de Bruijn sequences. *J. Combin. Theory Ser. A 22*, 1 (1977), 17 – 30.

18 FREDRICKSEN, H., AND MAIORANA, J. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Math. 23* (1978), 207–210.

19 GABRIĆ, D., AND SAWADA, J. A de Bruijn sequence construction by concatenating cycles of the complemented cycling register. In *Combinatorics on Words - 11th International Conference, WORDS 2017, Montréal, QC, Canada, September 11-15, 2017, Proceedings* (2017), pp. 49–58.

20 GABRIĆ, D., AND SAWADA, J. Constructing de Bruijn sequences by concatenating smaller universal cycles. *Theoretical Computer Science 743* (2018), 12–22.

21 GABRIĆ, D., AND SAWADA, J. Construction of orientable sequences in $O(1)$-amortized time per bit, https://arxiv.org/abs/2401.14341, 2024.

22 GABRIĆ, D., AND SAWADA, J. Efficient construction of long orientable sequences. In *35th Annual Symposium on Combinatorial Pattern Matching (CPM 2024)* (2024), pp. 1–12.

23 GABRIĆ, D., SAWADA, J., WILLIAMS, A., AND WONG, D. A framework for constructing de Bruijn sequences via simple successor rules. *Discrete Mathematics 241*, 11 (2018), 2977–2987.

24 GABRIĆ, D., SAWADA, J., WILLIAMS, A., AND WONG, D. A successor rule framework for constructing k -ary de Bruijn sequences and universal cycles. *IEEE Transactions on Information Theory 66*, 1 (2020), 679–687.

25 HIERHOLZER, C. Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. *Mathematische Annalen 6* (1873), 30–32.

26 HIGGNS, Z., KELLEY, E SIEBEN, B., AND GODBOLE, A. Universal and near-universal cycles of set partitions. *Electronic Journal of Combinatorics 22*, 4 (2015), P4.48.

27 HOLROYD, A. E., RUSKEY, F., AND WILLIAMS, A. Shorthand universal cycles for permutations. *Algorithmica 64*, 2 (2012), 215–245.

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

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

30 JACKSON, B. W. Universal cycles of k-subsets and k-permutations. *Discrete mathematics 117*, 1 (1993), 141–150.

31 JANSEN, C. J. A., FRANX, W. G., AND BOEKEE, D. E. An efficient algorithm for the generation of DeBruijn cycles. *IEEE Transactions on Information Theory 37*, 5 (Sep 1991), 1475–1478.

32 JOHNSON, J. R. Universal cycles for permutations. *Discrete Mathematics 309*, 17 (2009), 5264–5270.

33 KAK, S. Yamatarajabhanasalagam: an interesting combinatoric sutra. *Indian Journal of History of Science 35*, 2 (2000), 123–128.

34 Knuth, D. E. *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms*. Addison-Wesley Professional, 2011.

35 Mahadik, K., Wright, C., Kulkarni, M., Bagchi, S., and Chaterji, S. Scalable genome assembly through parallel de Bruijn graph construction for multiple k-mers. *Scientific Reports 9*, 1 (Oct 2019), 14882.

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

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

38 Ruskey, F., and Williams, A. An explicit universal cycle for the $(n$-$1)$-permutations of an n-set. *ACM Trans. Algorithms 6*, 3 (July 2010), 1–12.

39 Sala, E., Sawada, J., and Alhakim, A. Efficient constructions of the prefer-same and prefer-opposite de Bruijn sequences, https://arxiv.org/abs/2010.07960, 2020.

40 Sawada, J., Stevens, B., and Williams, A. De Bruijn sequences for the binary strings with maximum specified density. In *Proceeding of 5th International Workshop on Algorithms and Computation (WALCOM 2011), New Dehli, India, LNCS* (2011).

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

42 Sawada, J., Williams, A., and Wong, D. Universal cycles for weight-range binary strings. In *Combinatorial Algorithms - 24th International Workshop, IWOCA 2013, Rouen, France, July 10-12, 2013, LNCS 8288* (2013), pp. 388–401.

43 Sawada, J., Williams, A., and Wong, D. The lexicographically smallest universal cycle for binary strings with minimum specified weight. *Journal of Discrete Algorithms 28* (2014), 31–40.

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

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

46 Sawada, J., and Wong, D. Efficient universal cycle constructions for weak orders. *Discrete Mathematics 343*, 10 (2020), 112022.

47 Song, L., Geng, F., Gong, Z.-Y., Chen, X., Tang, J., Gong, C., Zhou, L., Xia, R., Han, M.-Z., Xu, J.-Y., Li, B.-Z., and Yuan, Y.-J. Robust data storage in DNA by de Bruijn graph-based de novo strand assembly. *Nature Communications 13*, 1 (Sep 2022), 5361.

48 Stein, S. K. The mathematician as an explorer. *Scientific American 204*, 5 (1961), 148–161.

49 Stein, S. K. *Mathematics: the man-made universe*. Courier Corporation, 2013.

50 Van Nooten, B. Binary numbers in Indian antiquity. *Journal of Indian philosophy* (1993), 31–50.

51 Wong, D. A new universal cycle for permutations. *Graph. Comb. 33*, 6 (Nov. 2017), 1393–1399.

52 Zerbino, D. R., and Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res. 18*, 5 (May 2008), 821–829.