Deduplication on Finite Automata and Nested Duplication Systems

Da-Jung Cho, Yo-Sub Han(✉), and Hwee Kim

Department of Computer Science, Yonsei University, 50 Yonsei-Ro, Seodaemun-Gu, Seoul 03722, Republic of Korea
{dajungcho,emmous,kimhwee}@yonsei.ac.kr

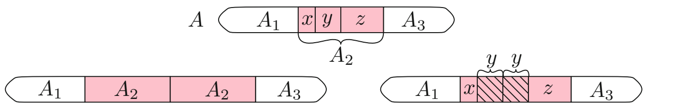
Abstract. Motivated by work on bio-operations on DNA sequences, a string duplication system S consists of an initial string over Σ and a set of duplication functions that iteratively generate new strings from existing strings in the system. As the main result we introduce the concept of a deduplication—a reverse function of duplication—on an nondeterministic finite-state automaton (NFA) and propose the deduplication operation on an NFA that transforms a given NFA to a smaller NFA while generating the same language in the string duplication system. Then, we introduce a nested duplication, which is similar to tandem duplication but depends on the information of the nested duplication in the previous step. We propose an NFA construction for an arbitrary nested duplication system, analyze its properties and present an algorithm that computes the system capacity.

Keywords: Bio-inspired operations · Tandem duplication · String duplication systems · Capacity · Automata theory

1 Introduction

A tandem duplication—a subsequence of a DNA sequence is placed next to the original position of a DNA sequence—occurs during DNA replication, indeed, more than half of the human genome consists of repetitive sequences [4]. Tandem duplications are well-studied in both DNA computing and formal language theory. From a formal language framework, a tandem duplication of a string xyz generates a new string $xyyz$, where $x, y, z \in \Sigma^*$. Many researchers [1–3,7,8,11, 13,15–17,20] considered the duplication operation and investigated their properties under formal language theory. Searls [17] introduced a formal representation of DNA recombination events. Dassow et al. [2,3] and Ito et al. [7] considered the duplication operation and investigated their closure properties of languages in the Chomsky hierarchy. Several researchers [8,11,13,14] considered (uniformly-) bounded duplication, which has an restriction on the length of duplicated factor and investigated closure properties with several conditions on the size of alphabet and length of duplicated factor. Both Martín-Vide and Păun [15] and Mitrana and Rozenberg [16] investigated the generative power of context-free and context-sensitive duplication grammars. Recently, Cho et al. [1] defined an extended variant of duplication and investigated their closure properties.

From an information theory viewpoint, Jain et al. [9] and Farnoud et al. [5] investigated the *string duplication system* that generates all strings obtained by applying the duplication function to an initial string a finite number of times, and computed the exact or bounded capacity of duplication systems considering several different alphabets, initial string and length of duplication. They suggested a concept of the *deduplication operation*, which is a reverse function of a duplication operation, on a regular expression.



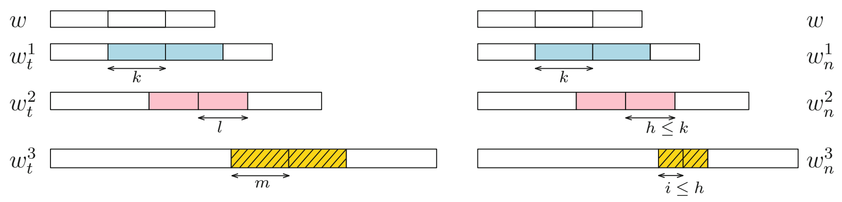
(a) A partial tandem duplication (b) An internal tandem duplication

Fig. 1. Given a gene sequence $A = A1A2A3$, (a) an internal tandem duplication duplicates a subsequence y generating $A1xyyzA3$, internal to $A2$, (b) a partial duplication duplicates a junction $A2$ to the end of $A2$.

In molecular biology, a tandem duplication is classified to a *partial tandem duplication* (PTD) or an *internal tandem duplication* (ITD) [18]. Figure 1 shows an example of partial and internal tandem duplications on a gene. The internal tandem duplication has an restriction on the position of duplicated factor, and it motivates us to define a *nested duplication*, which can be viewed as a generalization of an internal tandem duplication. A nested duplication considers the position and the size of the duplication segment of the previous duplications while general tandem duplications occur independently. Figure 2 shows that when duplication segments overlap each other, the size of nested duplication segments decreases over time whereas tandem duplication segments do not. Note that the size of a nested duplication segment is dependent on the nested duplication segment of the previous duplication step.

We can compute the capacity of a language L using Perron-Frobenius theory [6] if L is regular, or Chomsky-Schützenberger enumeration theorem [10] if L is unambiguously context-free. However, the language generated from a tandem duplication system turns out to be context-free in general cases [14]. On the other hand, a nested duplication system with an arbitrary seed and (fixed) duplication lengths always gives a regular language L even with the expanded cases of seed and duplication length.

We suggest the concept of deduplication on NFAs that reduces the size of an NFA while maintaining the same resulting language from the duplication system and give a formal construction, which was implicit in the previous research [9]. Then, we suggest an NFA construction that recognizes the language generated from a nested duplication system, and prove that there exists the case when the constructed NFA is the minimal DFA. We also propose an algorithm that computes the capacity of a nested duplication system using the construction.



(a) overlapping tandem duplications on w (b) overlapping nested duplications on w

Fig. 2. An example of overlapping tandem and nested duplications on an initial string w of length n. Let w_t^i be the string generated by the ith tandem duplication from w, and w_n^i be the string generated by the ith nested duplication from w, where $i = \{1, 2, 3\}$. (a) For a string w, a tandem duplication segment of length k is duplicated w_t^1, where $k \leq n$. The second and third duplications, w_t^2 and w_t^3, show that a duplication segment that overlaps the previous duplication segment is not dependent to the size of previously duplicated segment. (b) A nested duplication segment of length k is duplicated w_n^1, where $k \leq n$. The second and the third nested duplications have a smaller fragment size than the first and the second duplications, respectively. In other words, let h, i be the size of the second and the third duplication fragment, respectively. Then, $1 \leq i \leq h \leq k \leq n$.

2 Preliminaries

The reader may refer to Wood [19] for more knowledge in finite automata and formal languages. The symbol Σ denotes a finite alphabet, Σ^* is the set of strings over Σ, $|w|$ is the length of a string $w \in \Sigma^*$ and λ is the empty string. A string $x = x[1]x[2] \cdots x[n] \in \Sigma^n$ is a finite sequence of n symbols over $x[i] \in \Sigma$. We denote a string $x = x[1] \cdots x[n]$ by $x[1 : n]$ according to indices of each characters of x. For two strings $x = x[1 : n]$ and $y = y[1 : m]$ we denote the *catenation* of x and y by $x[1 : n]{\cdot}y[1 : m]$. We use w^n to denote the string resulted from catenation of n consecutive w's. We refer to a string $x \in \mathbb{N}^n$ as a *natural number array*, or simply an array, where $\mathbb{N}$ is the set of non-negative integers. A nondeterministic finite-state automaton (NFA) is a tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of states, Σ is the alphabet, $\delta \subseteq Q \times \Sigma \to 2^Q$ is the multivalued transitions function, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. Note that we can regard states as vertices and $\delta(p, a) = q$ as a labeled directed edge between two vertices p and q. Let $|Q|$ be the number of states in Q and $|\delta|$ be the number of transitions in δ. Then, the size of M is $|M| = |Q| + |\delta|$. Given a transition $\delta(p, a) = q$, we say that q has an *in-transition* and p has an *out-transition*. A string w is accepted by A if there is a labeled path from q_0 to a final state and we call this path an *accepting path*. The language accepted by M is $L(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$. The automaton M is a deterministic finite-state automaton (DFA) if δ is a single valued partial function. A sequence $q_0, q_1, \cdots, q_n$ of states denotes a *path*, and a path has a *cycle* if it has an occurrence of the same state twice in the path. We define the *size* of a cycle by the number of distinct states in the cycle.

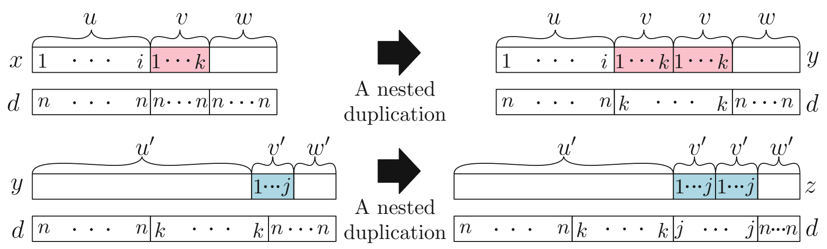


Fig. 3. An example of nested duplications on $x = uvw$, where $|u| = i, |v| = k, |x| = n$, and $d[1 : n] = n \cdots n$. A substring $v = x[i{+}1 : i{+}k]$ is duplicated if $d[i{+}1] \geq k$, which implies that $x = uvw$ transforms into $y = uvvw$ and the corresponding array d is updated as $d[i{+}1 : 2k] = k^{2k}$. From $y = u'v'w'$, a nested duplication of length j occurs on the position $i' + 1$ of y, where $|u'| = i', |v'| = j$ and $d[i' + 1] \geq j$. Then, the corresponding array $d[i'{+}1 : i'{+}j]$ is updated to j^{2j}.

A *tandem duplication* function $\mathbb{D}^{tan}_{i,k}$ is defined as follows:

$$\mathbb{D}^{tan}_{i,k}(x) = \begin{cases} uvvw & \text{for } x = uvw, |u| = i, |v| = k, \\ x & \text{otherwise.} \end{cases}$$

Note that $\mathbb{D}^{tan}_{i,k}(x)$ allows substring of length k starting at position $i + 1$ to be duplicated next to its original position. Similarly, $\mathbb{D}^{tan}_{\leq k}$ allows substring of length at most k to be duplicated.

We propose a new duplication function called the *nested duplication*. A nested duplication function has two inputs—a string x to duplicate and an array d indicating the duplication depth of each character in the string (See Fig. 3 for an example). The function $\mathbb{D}^{nes}_{i,k}$ is defined as follows:

$$\mathbb{D}^{nes}_{i,k}(x, d) = \begin{cases} (uvvw, d[1 : |u|] \cdot |v|^{2|v|} \cdot d[|u|{+}|v|{+}1 : |x|]) \\ \text{for } x = uvw, |u| = i, |v| = k, \\ d[j] \geq |v| \text{ for } |u| + 1 \leq j \leq |u| + |v|, \\ (x, d) \text{ otherwise.} \end{cases}$$

A *string duplication system* consists of three tuples $S = (\Sigma, s, \tau)$, where $s \in \Sigma^*$ is a finite length string called *seed*, an initial string of finite length, and τ is the set of rules that generate new strings from existing strings in the system [5]. We call the set of all strings generated by the system S *the language generated by the system*, and denote the language by $L(S)$. In the rest of paper, the nested duplication function $\mathbb{D}^{nes}_{\leq k}$ refers to the rule τ, and the term *nested duplication system* refers to the string duplication system over nested duplication function. We assume that the duplication length array for the seed is initialized as $|s|$ for all indices. We call a system $S = (\Sigma, s, \mathbb{D}^{nes}_{\leq k})$ a *bounded system* if $k \leq n$. We extend the nested duplication system to a language $S = (\Sigma, L, \mathbb{D}^{nes}_{\leq k})$ that generates all duplicated strings from a given initial language L. Note that $L(\Sigma, s, \mathbb{D}^{nes}_{\leq k}) \subseteq L(\Sigma, s, \mathbb{D}^{nes}_{\leq k})$. For example, let $S_1 = (\{a, b, c, d\}, abcd, \mathbb{D}^{nes}_{\leq 4})$ and

$S_2 = (\{a, b, c, d\}, abcd, \mathbb{D}^{tan}_{\leq 4})$. A string $abcdacadacacdabcd$ is in $L(S_2)$ but not in $L(S_1)$. In S_2, the system can generate the string by a sequence of duplications $abcd \rightarrow abcdabcd \rightarrow abcdacdabcd \rightarrow abcdacacdabcd \rightarrow abcdacadacacdabcd$. However, the last duplication is not possible in S_1 due to the condition of the duplication depth array. Namely, the last duplication duplicates a substring of length 4 within the duplicated substring of length 2, which is not allowed in a nested duplication system.

The *capacity* of a duplication system S represents how many strings the system produces compared to Σ^n, where n goes to infinity, and it is defined by

$$\text{cap}(S) = \lim_{n\rightarrow\infty} \sup \frac{\log_{|\Sigma|} |S \cap \Sigma^n|}{n}.$$

3 $\mathbb{D}$-Cycle Deduplication on Finite Automata

A duplication (both tandem and nested) on a string $w = xyz$ allows a substring y to be duplicated next to its original position in w and generate a string $w' = xyyz$, where $x, y, z \in \Sigma^*$. A *deduplication* of length k is an operation that transforms a substring yy into y, where $|y| = k$. Namely, a deduplication of $w' = xyyz$ is $w = xyz$. The operation was defined in various names by researchers [7,12]. Note that both duplication systems with the same duplication rule and different seeds w' and w generates the same language, where $|w'| > |w|$. Based on these aspects, a deduplication on a language is defined as follows: Given a language L_1, $L_1 \overset{dd_{\leq k}}{\longrightarrow} L_2$ denotes a deduplication on L_1 when two duplication systems with duplication length up to k and two seeds L_1 and L_2 are same, and there is a metric $\mathbb{S}$ to measure the size of the language such that $\mathbb{S}(L_1) > \mathbb{S}(L_2)$. For instance, $L(a^+)$ can be transformed into $L(a)$ by $L(a^+) \overset{dd_{\leq 3}}{\longrightarrow} L(a)$ when the size of the language is measured by the number of states of the minimal DFA that recognizes the language.

Though a deduplication on a string is straightforward, it is hard to do a deduplication on infinite languages such as regular languages or context-free languages. Now, we introduce a special deduplication on an NFA called $\mathbb{D}$-*cycle deduplication* that transforms a given NFA to a smaller NFA while generating the same language in the duplication system by removing cycles in the NFA that satisfies special conditions. For the following description, we assume that a given NFA has no λ-transition and all states are reachable. For a cycle C in an NFA, we call the cycle $\mathbb{D}$-cycle if C satisfies the following conditions:

(i) The cycle C is defined by a sequence of states $(q_{in}, q'_1, q'_2, \ldots, q'_i, q_{out}, q_1, q_2, \ldots, q_j, q_{in})$, where $i, j \geq 0$. All states except q_{in} and q_{out} have one in-transition from the previous state and one out-transition to the following state. The state q_{in} has more than one in-transition, which implies that there exists an in-transition from a state outside of the cycle. The state q_{out} has also more than one out-transition.

(ii) There exist positive integers k and l that satisfy the following conditions:
  (a) All paths of size $k+1$ from q_{out} yield a same set of strings.
  (b) All paths of size $l+1$ to q_{in} yield a same set of strings.
  (c) $k+l = j+1$.

We are now ready to introduce $\mathbb{D}$-cycle deduplication on an NFA (See Fig. 4 for an example).

Definition 1. *Given an NFA $M = (Q, \Sigma, \delta, s, F)$ with a $\mathbb{D}$-cycle $(q_{in}, q'_1, q'_2, \ldots, q'_i, q_{out}, q_1, q_2, \ldots, q_j, q_{in})$, we define a $\mathbb{D}$-cycle deduplication by $M \xrightarrow{\mathbb{D}^{-1}_{\leq h}} M'$, where $i + j = h$ to be*

$$M' = (Q \setminus \{q_1, q_2, \ldots, q_j\}, \Sigma, \{\delta(p, \sigma) \to q \mid p, q \notin \{q_1, q_2, \ldots, q_j\}\}, s, F).$$

We say that the cycle $(q_{in}, q'_1, q'_2, \ldots, q'_i, q_{out}, q_1, q_2, \ldots, q_j, q_{in})$ is removed from M.

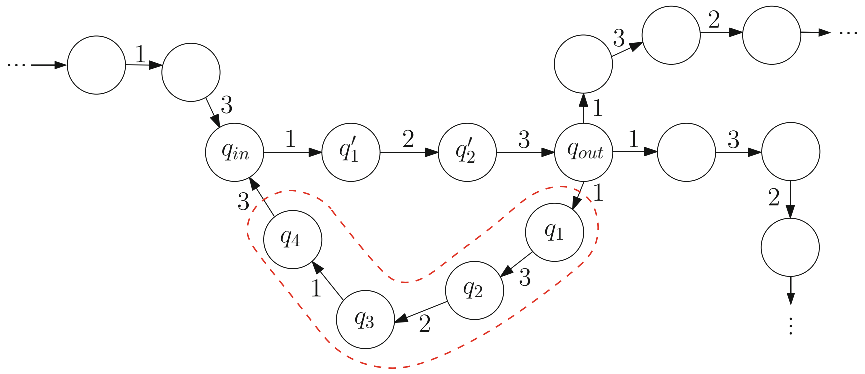


Fig. 4. An example of a $\mathbb{D}$-cycle $(q_{in}, q'_1, q'_2, q_{out}, q_1, q_2, q_3, q_4, q_{in})$. All states of the cycle except q_{in} and q_{out} have one in-transition from the previous state and one out-transition to the following state. All paths of size 4 from q_{out} yield the same string 132 and all paths of size 3 to q_{in} yield the same string 13. Also, the size of the path from q_{out} to q_{in} in the cycle is 6. Then, all states and transitions in the dashed line can be eliminated.

We use $\mathcal{S}(L(M)) = |M| = |Q| + |\delta|$. By this measure, it is straightforward that the size of the automaton is reduced. We now prove that the resulting automaton generates the same language in the tandem duplication system. Let $S^{tan}_{\leq h}(A) = (\Sigma, L(A), \mathbb{D}^{tan}_{\leq h})$ be the tandem duplication system for an NFA M.

Lemma 1. *Given an NFA $M = (Q, \Sigma, \delta, s, F)$ and its deduplication M' such that $M \xrightarrow{\mathbb{D}^{-1}_{\leq h}} M'$, $L(S^{tan}_{\leq h}(M)) = L(S^{tan}_{\leq h}(M'))$.*

From Lemma 1, we prove that $\mathbb{D}$-cycle deduplication satisfies the conditions of deduplication on a language. Using $\mathbb{D}$-cycle deduplication, we can reduce the size of the seed regular language for a tandem duplication system.

4 An NFA Construction for a Nested Duplication System

We introduce an NFA construction for a nested duplication system with an arbitrary seed and (fixed) duplication lengths. First, we show that the construction can be applied to a system with a designated set of duplication lengths, not only for a bounded system. Second, we show that the construction can be applied to a system with the maximum duplication length greater than the length of the seed. Then, we give a procedure of computing the capacity of a nested duplication system.

We first introduce an NFA construction for a nested duplication system with an arbitrary seed s and duplication length up to $k = |s|$. For a bounded nested duplication system $S = (\Sigma, s, \mathbb{D}^{nes}_{\leq n})$ where $|s| = n$, we define an NFA $M_S = (Q, \Sigma, \delta, 0_0, \{n_0\})$ as follows: The set of states is defined as $Q = \{q = l[1]_{d[1]}l[2]_{d[2]} \cdots l[t]_{d[t]} \mid 1 \leq t \leq n, d[1] = 0, d[i] = n - i + 2, l[i] < d[i]$ for $2 \leq i \leq n\}$, and transitions are defined by the following.

(i) For a pair of states $(i{-}1)_0$ and i_0 where $1 \leq i \leq n$,
  (a) $\delta((i{-}1)_0, s[i]) = i_0$.
  These transitions make the path for the seed.
  (b) $\delta(2_0, s[1]) = 1_0$.
  This transition makes a cycle of size 2 for the duplication of $s[1:2]$.
  (c) $\delta(i_0, s[1]) = i_0 1_i$.
  (d) $\delta(i_0(j{-}1)_i, s[j]) = i_0 j_i$ for $1 \leq j \leq i - 2$.
  (e) $\delta(i_0(i{-}2)_i, s[i{-}1]) = (i{-}1)_0$.
  These transitions make a cycle of size i for the duplication of $s[1:i]$.
(ii) For a state $q = l[1]_0 l[2]_{d[2]} \cdots l[t]_{d[t]}$, let the string $y(q)$ be recursively defined as follows:
  (a) For $i = 1$, $y(q)$ is initialized as $s[1 : l[1]]$.
  (b) While increasing i up to t, let new $y(q)$ be $y(q) \cdot y(q)[|y(q)|{-}d[i]{+}1 : |y(q)|{-}d[i]{+}l[i]]$.
  For example, if $s = abcd$ and $q = 4_0 2_4 1_3$, then $y(q) = abcdabd$.
  Then, for an integer $d' \leq n$,
  (a) $\delta(q, y(q)[|y(q)|{-}d'{+}1]) = l[1]_0 l[2]_{d[2]} \cdots l[t]_{d[t]} 1_{d'}$ when $d' < d[t]$ or $t = 1$.
  (b) $\delta(l[1]_0 l[2]_{d[2]} \cdots l[t]_{d[t]}(j{-}1)_{d'}, y(q)[|y(q)|{-}d'{+}j]) = l[1]_0 l[2]_{d[2]} \cdots l[t]_{d[t]} j_{d'}$ for $2 \leq j \leq d' - 1$.
  (c) $\delta(l[1]_0 l[2]_{d[2]} \cdots l[t]_{d[t]}(d'{-}1)_{d'}, y(q)[|y(q)|]) = q$.
  These transitions make a cycle of size $d' < d[t]$ for the duplication of a string $y(q)$.
  (d) $\delta(q, y(q)[|y(q)|]) = q$ except for $q = 0_0$.
  These transitions make a self loop for the duplication of single character.

We give a description for the semantics of the states. For a state $q = l[1]_{d[1]}l[2]_{d[2]} \cdots l[t]_{d[t]}$, t represents the depth of a duplication. For each depth i, there can be arbitrary many duplications of length $d[i]$, and $l[i]$ represents how many characters are so far duplicated by the last duplication in depth i. Note that $d[i] > d[i{+}1]$ for $i > 1$, which indicates that the duplication length decreases as the depth increases. This property indicates that the automaton accepts the language generated from a nested duplication system, not a tandem duplication system. The first length $d[1]$ is always 0, which indicates the original seed. Once the construction is given, $y(q)$ is equal to the string that is yielded by the shortest path from the start state. For a state q, we define the *largest duplication length (LDL)* to be the size of the largest simple cycle that includes the state. If $q = l[1]_0$, then $LDL(q) = l[1] + 1$ except for the start state, where $LDL(0_0) = 0$. In all other cases, $LDL(q) = d[t]$ (See Fig. 5 for an example).

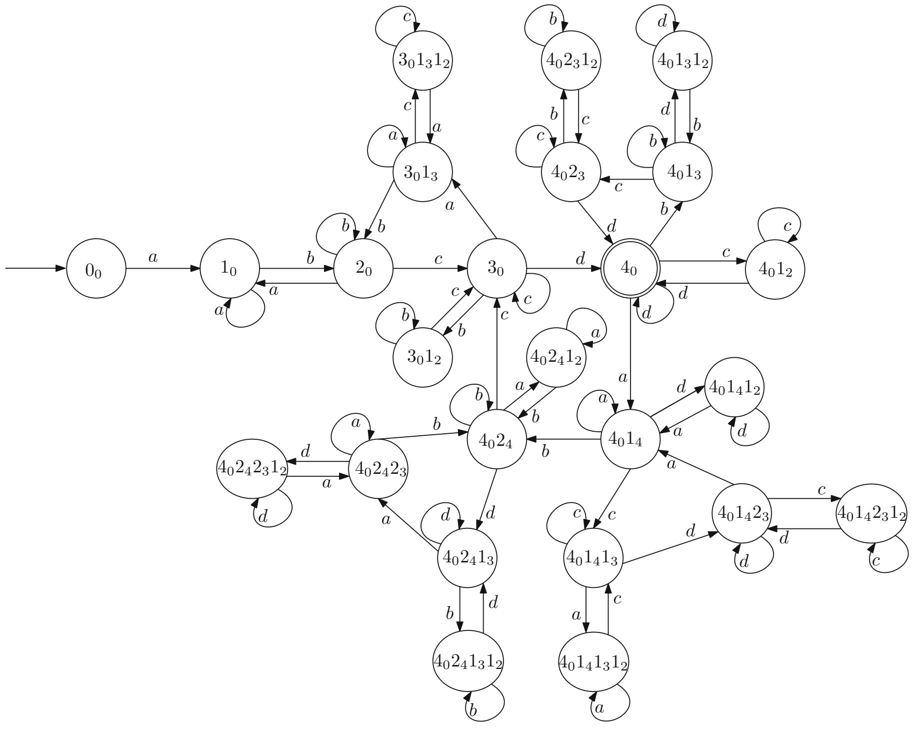


Fig. 5. An example of an NFA $M_S = (Q, \Sigma, \delta, 0_0, \{4_0\})$ for $S = (\Sigma, abcd, \mathbb{D}^{nes}_{\leq 4})$. Note that $y(4_02_41_3) = abcdabd$, which is equal to the string that is yielded by the shortest path from the start state. Also note that $LDL(3_0) = 4$ and $LDL(4_01_41_3) = 3$.

Theorem 1. *For a string s of length n, the NFA $M_S = (Q, \Sigma, \delta, 0_0, \{n_0\})$ recognizes the language generated by the nested duplication system $S = (\Sigma, s, \mathbb{D}^{nes}_{\leq n})$. Namely, $L(M_S) = L(S)$.*

Proof. We first prove that if a string $x \in L(S)$, then $x \in L(M_S)$. We start from the following claim:

Claim. For a state q in Q, let y be a string yielded by a path of length less than or equal to $LDL(q)$ that ends at q. Then there always exists a cyclic path from q to q that yields y.

We have two observations from the construction: First, for each string y yielded by a non-cyclic path p of length less than or equal to $LDL(q)$ to q, there always exists a cyclic path p' from q to q that yields y. Second, for each string y' yielded by a cyclic path p_c of length less than or equal to $LDL(q)$ with a state in p, there always exists a cyclic path p'_c that yields y with a state in p'. From two observations, we know that the claim holds.

Now, we use induction on the duplication step of x. We add another claim to the theorem: For a string x, let the path $(q_0, q_1, \ldots, q_{|x|})$ be the path that yields x and d_x be the duplication length array for x. Then, for all $1 \leq i \leq |x|$, $d_x[i] \leq LDL(q_i)$.

Base Case. Suppose $x = s$. It implies that $x \in L(M_S)$. For all $1 \leq i \leq |x|$, $d_x[i] = LDL(q_i) = n$.

Inductive Step. Suppose for a string $x = uvz \in S$, $x \in L(M_S)$ and $d_x[i] \leq LDL(q_i)$ for all x with k duplication steps. Now, assume that $x' = uvvz \in S$ is duplicated from x and v is yielded by a path that ends at the state p. Since $d_x[i] \leq LDL(q_i)$ for all $1 \leq i \leq |x|$, $|v| \leq LDL(p)$. Thus, from the previous claim, we know that there exists a cyclic path from p to p that yields v. Therefore, $x' = uvvz \in L(M_S)$.

Second, we prove that if a string $x \in L(M_S)$, then $x \in L(S)$. For a string $x \in L(M_S)$, let p be the path that yields x. We recursively generate a series of paths p_i and strings x_i as follows: The path p_1 is the string generated by removing all self loops in p, and the path p_i is the string generated by removing all cycles of size i in p_{i-1}. The string x_i is yielded by the path p_i. From the construction, it is straightforward that $x_n = s$. Now, let d_i be an array of size $|x_i|$ and $d_n = n^n$. Then, we can successfully duplicate x_i from x_{i+1} in the nested duplication system S, since i represents the duplication length. Therefore, $x_i \in L(S)$ for $1 \leq i \leq n$ and $x \in L(S)$. □

Theorem 2. *If all characters in a string s of length n are distinct, then the NFA M_S constructed from the nested duplication system $S = (\Sigma, s, \mathbb{D}^{nes}_{\leq n})$ is indeed the minimal DFA with $n! + 1$ states and $en\Gamma(n, 1) + 1$ transitions, where Γ is the partial Gamma function.*

Proof. Let s be a string where $s[i] \neq s[j]$ if $i \neq j$. Note that the number of out-transitions from a state q is equal to $LDL(q)$. Thus, if $LDL(p) \neq LDL(q)$, then p and q are distinct. We also observe that for any simple cycle in M_S, all transitions in the cycle have pairwise distinct labels. Thus, any pair of states in a simple cycle (with the same LDL) is distinct. Now, suppose there are two states p and q where $LDL(p) = LDL(q)$, p and q are in a different simple cycle and p and q are equivalent. Then there should be two paths from p and q that end at the same state r and yields the same string. Since all characters in s are pairwise distinct, states before r in two paths should have different LDLs. Then, there exists two paths from p and q that end at distinct states and yield the same string. Thus, p and q are distinct.

We now compute the number of states and transitions in M_S by induction on a given n. We call the constructed DFA $M_S(n)$.

Base Case. For $n = 2$, $|Q| = 3 = 2! + 1$ and $|\delta| = 5 = 2e \times \frac{2}{e} + 1$.

Inductive Step. Assume that the statement holds for $n = k$. For $n = k + 1$, there exists one simple cycle of size $k + 1$ in the DFA, where each state in the cycle has the same topologic structure as $M_S(k)$ without 0_0. Therefore, the number of the states becomes $(k + 1)(k! + 1 - 1) + 1 = (k + 1)! + 1$, and the number of the transitions becomes $(k + 1)(ek\Gamma(k, 1) + 1 - 1) + (k + 1) + 1 = (k + 1)e(k\Gamma(k, 1) + \frac{1}{e}) - (k + 1) + (k + 1) + 1 = (k + 1)e\Gamma(k + 1, 1) + 1$. □

Theorem 2 shows that there exists a case where M_S is the minimal DFA that recognizes $L(S)$. Note that for all other cases, where there exist i, j such that $w[i] = w[j]$, M_S is an NFA.

The construction of M_S for the system S can be generalized to the following cases:

1. The seed can be given as a finite seed language. Since L is finite, we can construct NFAs for all strings in L and make an union NFA for S.
2. The bound k can be generalized to any positive integer. When $k \leq |s|$ and all characters in a string s of length n are distinct, the NFA M_S becomes the minimal DFA with $k!(n = k + 1) + 1$ states and $ek(n - k + 1)\Gamma(k, 1) + n - k + 1$ transitions, where n is the length of the seed.
3. The duplication length can be given as a set $\mathcal{L}$ of possible duplication lengths, not an inequality. Suppose $k_i \leq n$ and $k_{i-1} < k_i$ for $L = \{k_i\}$, $2 \leq i \leq m = |L|$. If all characters in a string s of length n are distinct, the NFA M_S becomes the minimal DFA with $(n - k_m + 1)k_m \prod_{i=1}^{m-1}(k_i - 1) + \sum_{i=1}^{m-1}\left(k_i\alpha_i \prod_{j=1}^{i-1}(k_j - 1)\right) + k_1 - 1$ states and $k_m(n - k_m + 1) + \sum_{i=1}^{m-1}\left(k_i\left((n - k_m + 1)k_m \prod_{j=1}^{i-1}(k_j - 1) + \sum_{j=1}^{i-1}\left(k_j\alpha_j \prod_{l=1}^{j-1}(k_l - 1)\right) + \alpha_i\right)\right) + n - m$ transitions, where

$$\alpha_i = \begin{cases} 1 & \text{if} k_{i+1} - k_i > 1, \\ 0 & \text{otherwise.} \end{cases}$$

Using the construction, we can design an algorithm to compute the capacity of S. Once the NFA M_S is generated, we first construct an equivalent DFA by the subset construction, and modify the DFA so that the DFA has at most one transition for every pair of states. This modification can be done by making copies of states that have multiple in-transitions from a state. From the resulting DFA, we can compute the capacity of the language recognized by the DFA using Perron-Frobenius Theory [6,9].

Procedure. ComputeCap(s)

```
1 Construct an NFA $M_S$ for $S = (\Sigma, s, \mathbb{D}^{tan}_{\leq n})$. /* $s$ is a seed of length $n$ */
2 Convert $M_S$ to a DFA $M'$, where there exists at most one transition for every pair of states.
3 Find the maximal connected component in $M'$ and compute its adjacency matrix $\mathbb{M}$.
4 Return the maximum eigenvalue of $\mathbb{M}$.
```

5 Conclusions

We have suggested the concept of deduplications—removing duplications—in an NFA and proposed an example of deduplications. Then, we have proposed a new duplication operation called the nested duplication: The operation is motivated by the internal tandem duplication phenomenon, which restricts the duplication length and position according to the previous duplication step. For a nested duplication system, we proposed an NFA construction that recognizes the system, and proved that this construction can be expanded to systems with various conditions. Using the constructed NFA, we have also proposed a procedure to compute the capacity of the given nested duplication system.

Although the concept of deduplications in an NFA is proposed, it is still open to find more duplication conditions applicable in an NFA. Finding deduplication conditions for other classes of FA, including pushdown automata, is one of the future works. For the nested duplication system, the tight bound of the size of the minimal DFA is an open problem. Inspection of subword closure properties on a nested duplication system is also one of the possible future works.

References

1. Cho, D.-J., Han, Y.-S., Kim, H., Palioudakis, A., Salomaa, K.: Duplications and pseudo-duplications. Int. J. Unconv. Comput. **12**, 145–167 (2016)
2. Dassow, J., Mitrana, V., Paun, G.: On the regularity of duplication closure. Bull. EATCS **69**, 133–136 (1999)
3. Dassow, J., Mitrana, V., Salomaa, A.: Operations and language generating devices suggested by the genome evolution. Theoret. Comput. Sci. **270**(1–2), 701–738 (2002)
4. de Koning, A.J., Gu, W., Castoe, T.A., Batzer, M.A., Pollock, D.D.: Repetitive elements may comprise over two-thirds of the human genome. PLoS Genet. **7**(12), e1002384 (2011)
5. Farnoud, F., Schwartz, M., Bruck, J.: The capacity of string-duplication systems. IEEE Trans. Inf. Theory **62**(2), 811–824 (2016)
6. Immink, K.: Codes for Mass Data Storage Systems. Shannon Foundation Publishers, Denver (2004)
7. Ito, M., Kari, L., Kincaid, Z., Seki, S.: Duplication in DNA sequences. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 419–430. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85780-8_33

8. Ito, M., Leupold, P., Shikishima-Tsuji, K.: Closure of language classes under bounded duplication. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 238–247. Springer, Heidelberg (2006). doi:10.1007/11779148_22
9. Jain, S., Farnoud, F., Bruck, J.: Capacity and expressiveness of genomic tandem duplication. In: Proceedings of the 23rd IEEE International Symposium on Information Theory, pp. 1946–1950 (2015)
10. Kuich, W., Salomaa, A.: Semirings, Automata and Languages. Springer, New York, Inc. (1985)
11. Leupold, P.: Languages generated by iterated idempotencies. Ph.D. thesis, University Rovira i Virgili (2006)
12. Leupold, P.: Duplication roots. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 290–299. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73208-2_28
13. Leupold, P., Martín-Vide, C., Mitrana, V.: Uniformly bounded duplication languages. Discret. Appl. Math. **146**(3), 301–310 (2005)
14. Leupold, P., Mitrana, V., Sempere, J.M.: Formal languages arising from gene repeated duplication. In: Jonoska, N., Păun, G., Rozenberg, G. (eds.) Aspects of Molecular Computing. LNCS, vol. 2950, pp. 297–308. Springer, Heidelberg (2003). doi:10.1007/978-3-540-24635-0_22
15. Martín-Vide, C., Păun, G.: Duplication grammars. Acta Cybern. **14**(1), 151–164 (1999)
16. Mitrana, V., Rozenberg, G.: Some properties of duplication grammars. Acta Cybern. **14**(1), 165–177 (1999)
17. Searls, D.B.: The computational linguistics of biological sequences. Artif. Intell. Mol. Biol. **2**, 47–120 (1993)
18. Swanson, L., Robertson, G., Mungall, K.L., Butterfield, Y.S., Chiu, R., Corbett, R.D., Docking, T.R., Hogge, D., Jackman, S.D., Moore, R.A., et al.: Barnacle: detecting and characterizing tandem duplications and fusions in transcriptome assemblies. BMC Genom. **14**(1), 550 (2013)
19. Wood, D.: Theory of Computation. Wiley, New York (1987)
20. Yokomori, T., Kobayashi, S.: DNA evolutionary linguistics, RNA structure modeling: a computational approach. In: Proceedings of the 1st International Symposium on Intelligence in Neural and Biological Systems, pp. 38–45 (1995)