Verifying a Parameterized Border Array in $O(n^{1.5})$ Time

Tomohiro I[1], Shunsuke Inenaga[2], Hideo Bannai[1], and Masayuki Takeda[1]

[1]Department of Informatics, Kyushu University
[2]Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishiku, Fukuoka, 819–0395 Japan.
tomohiro.i@i.kyushu-u.ac.jp
inenaga@c.csce.kyushu-u.ac.jp
{bannai,takeda}@inf.kyushu-u.ac.jp

Abstract. The parameterized pattern matching problem is to check if there exists a renaming bijection on the alphabet with which a given pattern can be transformed into a substring of a given text. A *parameterized border array* (*p-border array*) is a parameterized version of a standard border array, and we can efficiently solve the parameterized pattern matching problem using p-border arrays. In this paper we present an $O(n^{1.5})$-time $O(n)$-space algorithm to verify if a given integer array of length n is a valid p-border array for an unbounded alphabet. The best previously known solution takes time proportional to the n-th Bell number $\frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$, and hence our algorithm is quite efficient.

1 Introduction

The *parameterized matching* (*p-matching*) problem [1] is a kind of string matching problem, where a pattern is considered to occur in a text when there exists a renaming bijection on the alphabet with which the pattern can be transformed into a substring of the text. Parameterized matching has applications in e.g. software maintenance, plagiarism detection, and RNA structural matching, thus it has extensively been studied (e.g., see [2–6]).

In this paper we focus on *parameterized border arrays* (*p-border arrays*) [7], which are a parameterized version of border arrays [8]. Let Π be the alphabet. The p-border array of a given pattern p of length m can be computed in $O(m \log |\Pi|)$ time, and the p-matching problem can be solved in $O(n \log |\Pi|)$ time for any text p-string of length n, using the p-border array [7].

This paper deals with the *reverse engineering problem on p-border arrays*, namely, the problem of verifying if a given integer array of length n is a p-border array of some string. We propose an $O(n^{1.5})$-time $O(n)$-space algorithm to solve this problem for an unbounded alphabet. We emphasize that the best previously known solution to this problem takes time proportional to the n-th Bell number $\frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$, and hence our algorithm is quite efficient.

Related Work. There exists a linear time algorithm to solve the reverse problem on p-border arrays for a binary alphabet [9]. An $O(p_n)$-time algorithm to enumerate all p-border arrays of length up to n on a binary alphabet was also presented in [9], where p_n denotes the number of p-border arrays of length at most n for a binary alphabet.

In [10], a linear time algorithm to verify if a given integer array is the (standard) border array [8] of some string was presented. Their algorithm works for both bounded and unbounded alphabets. A simpler linear-time solution for the same problem for a bounded alphabet was shown in [11]. An algorithm to enumerate all border arrays of length at most n in $O(b_n)$-time was given in [10], where b_n is the number of border arrays of length at most n.

The reverse engineering problems, as well as the enumeration problems for other string data structures (suffix arrays, DAWG, etc.) have been extensively studied [12–18], whose solutions give us further insight concerning the data structures.

2 Preliminaries

Let Σ and Π be two disjoint finite alphabets. An element of $(\Sigma \cup \Pi)^*$ is called a *p-string*. The length of any p-string s is the total number of constant and parameter symbols in s and is denoted by $|s|$. The string of length 0 is called the empty string and is denoted by ε. For any p-string s of length n, the i-th symbol is denoted by $s[i]$ for each $1 \leq i \leq n$, and the *substring* starting at position i and ending at position j is denoted by $s[i:j]$ for $1 \leq i \leq j \leq n$.

Any two p-strings $s, t \in (\Sigma \cup \Pi)^*$ of length m are said to *parameterized match* (*p-match*) if s can be transformed into t by a renaming function f from the symbols of s to the symbols of t, where f is the identify on Σ. The p-matching problem on $\Sigma \cup \Pi$ is reducible in linear time to the p-matching problem on Π [2]. Thus we will only consider p-strings over Π.

Let $\mathcal{N}$ be the set of non-negative integers. Let $pv : \Pi^* \to \mathcal{N}^*$ be the function s.t. for any p-string s of length $n > 0$, $pv(s) = u$ where, for $1 \leq i \leq n$, $u[i] = 0$ if $s[i] \neq s[j]$ for any $1 \leq j < i$, and $u[i] = i - k$ if $k = \max\{j \mid s[i] = s[j], 1 \leq j < i\}$. Let $pv(\varepsilon) = \varepsilon$. Two p-strings s and t of the same length m p-match iff $pv(s) = pv(t)$. For any $p \in \mathcal{N}^*$, let $zeros(p)$ denotes the number of 0's in p, that is, $zeros(p) = |\{i \mid p[i] = 0, 1 \leq i \leq |p|\}|$. For any $s \in \Pi$, $zeros(pv(s))$ equals the number of different characters in s. For example, `aabb` and `bbaa` p-match since $pv(\texttt{aabb}) = pv(\texttt{bbaa}) = 0\ 1\ 0\ 1$. Note $zeros(pv(\texttt{aabb})) = zeros(pv(\texttt{bbaa})) = 2$.

A *parameterized border* (*p-border*) of a p-string s of length n is any integer j s.t. $0 \leq j < n$ and $pv(s[1:j]) = pv(s[n-j+1:n])$. For example, the set of p-borders of p-string `aabb` is $\{2, 1, 0\}$ since $pv(\texttt{aa}) = pv(\texttt{bb}) = 0\ 1$, $pv(\texttt{a}) = pv(\texttt{b}) = 0$, and $pv(\varepsilon) = pv(\varepsilon) = \varepsilon$. We also say that b is a p-border of $p \in \mathcal{N}^*$ if b is a p-border of some p-string $s \in \Pi^*$ and $p = pv(s)$. The *parameterized border array* (*p-border array*) β_s of a p-string s of length n is an array of length n such that $\beta_s[i] = j$, where j is the longest p-border of $s[1:i]$. For example, for p-string $s = \texttt{aabbaa}$, $\beta_s = [0, 1, 1, 2, 3, 4]$. When it is clear from the context, we abbreviate β_s as β. Let $\mathrm{P} = \{pv(s) \mid s \in \Pi^*\}$ and $\mathrm{P}_\beta = \{p \in \mathrm{P} \mid \beta[i]$ is the longest p-border of $p[1:i], 1 \le i \le |\beta|\}$.

For any $i, j \in \mathcal{N}$, let $cut(i,j) = 0$ if $i \ge j$, and $cut(i,j) = i$ otherwise. For any $p \in \mathrm{P}$ and $1 \le j \le |p|$, let $suf(p,j) = cut(p[|p|-j+1],1)cut(p[|p|-j+2],2)\cdots cut(p[|p|],j)$. Let $suf(p,0) = \varepsilon$. For example, if $p[1:10] = 0\ 0\ 2\ 0\ 3\ 1\ 3\ 2\ 6\ 3$,

$$\begin{aligned} suf(p,5) &= cut(p[6],1)cut(p[7],2)cut(p[8],3)cut(p[9],4)cut(p[10],5) \\ &= cut(1,1)cut(3,2)cut(2,3)cut(6,4)cut(3,5) = 0\ 0\ 2\ 0\ 3. \end{aligned}$$

Then, for any p-string $s \in \Pi^*$ and $1 \le j \le |s|$, $suf(pv(s),j) = pv(s[|s|-j+1:|s|])$. Hence, j is a p-border of $pv(s)$ iff $suf(pv(s),j) = pv(s)[1:j]$ for some $1 \le j < |s|$.

This paper deals with the following problem.

Problem 1 (Verifying a valid p-border array). Given an integer array y of length n, determine if there exists a p-string s such that $\beta_s = y$.

To solve Problem 1, we can use the algorithm of Moore et al. [19] to generate all strings in $\mathrm{P}^n = \{p \mid p \in \mathrm{P}, |p| = n\}$ in $O(|\mathrm{P}^n|)$ time, and then we check if $p \in \mathrm{P}_y$ for each generated $p \in \mathrm{P}^n$. Still, it is known that $|\mathrm{P}^n|$ is equal to the n-th Bell number $\frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!}$.

As a much more efficient solution, we present our $O(n^{1.5})$-time algorithm in the sequel.

3 Properties on Parameterized Border Arrays

Here we introduce important properties of p-border arrays that are useful to solve Problem 1.

For any integer array ℓ, let $|\ell|$ denote the length of the integer array ℓ. Let $\ell[i:j]$ denote a subarray of ℓ for any $1 \le i \le j \le |\ell|$. Let $\Gamma = \{\gamma \mid \gamma[1] = 0, 1 \le \gamma[i] \le \gamma[i-1]+1, 1 < i \le |\gamma|\}$. For any $\gamma \in \Gamma$ and any $i \ge 1$, let $\gamma^k[i] = \gamma[i]$ if $k = 1$, and $\gamma[\gamma^{k-1}[i]]$ if $k > 1$ and $\gamma^{k-1}[i] \ge 1$. By the definition of Γ, the sequence $i, \gamma^1[i], \gamma^2[i], \ldots$ is monotonically decreasing and terminates with $1, 0$. Let $\mathrm{A} = \{\alpha \mid \alpha \in \Gamma, \alpha[i] \in \{\alpha^1[i-1]+1, \alpha^2[i-1]+1, \ldots, 1\}, 1 < i \le |\alpha|\}$. It is clear that $\mathrm{A} \subset \Gamma$. Let B denote the set of all p-border arrays.

Lemma 1. $\mathrm{B} \subseteq \Gamma$.

Proof. By definition, it is clear that $\beta[1] = 0$ and $1 \le \beta[i]$ for any $1 < i \le |\beta|$. For any $p \in \mathrm{P}_\beta$ and i, since $suf(p[1:i],\beta[i]) = p[1:\beta[i]]$, $suf(p[1:i-1],\beta[i]-1) = p[1:\beta[i]-1]$. Thus $\beta[i-1] \ge \beta[i]-1$, and therefore $\beta[i] \le \beta[i-1]+1$. □

Lemma 2. *For any $\beta \in \mathrm{B}$, $p \in \mathrm{P}_\beta$, and $1 \le i \le |p|$, $\{\beta^1[i], \beta^2[i], \ldots, 0\}$ is the set of p-borders of $p[1:i]$.*

Lemma 3. *For any $\beta \in \mathrm{B}$, $p \in \mathrm{P}_\beta$, and $1 \le i \le |p|$, if $p[i] = 0$, then $p[b] = 0$ for any $b \in \{\beta^1[i], \beta^2[i], \ldots, 1\}$.*

Lemma 4. $\mathrm{B} \subseteq \mathrm{A}$.

Proof. For any $\beta \in \mathrm{B}, p \in \mathrm{P}_\beta$ and $1 < i \leq |p|$, since $\mathit{suf}(p[1:i], \beta[i]) = p[1:\beta[i]]$, $\mathit{suf}(p[1:i-1], \beta[i]-1) = p[1:\beta[i]-1]$. Since $\beta[i]-1$ is a p-border of $p[1:i-1]$, $\beta[i]-1 \in \{\beta^1[i-1], \beta^2[i-1], \ldots, 0\}$ by Lemma 2. Hence, $\beta[i] \in \{\beta^1[i-1]+1, \beta^2[i-1]+1, \ldots, 1\}$. □

Definition 1 (Conflict Points). *Let $\alpha \in \mathrm{A}$. For any c', c $(1 < c' < c \leq |\alpha|)$, if $\alpha[c'] = \alpha[c]$ and $c'-1 = \alpha^k[c-1]$ with some k, then c' and c are said to be* in conflict *with each other. Such points are called* conflict points.

Let C_α be the set of conflict points in α and $C_\alpha(c)$ be the set of points that conflict with c $(1 \leq c \leq |\alpha|)$. For any $i \leq j \in \mathcal{N}$, let $[i,j] = \{i, i+1, \ldots, j\} \subset \mathcal{N}$. We denote $C_\alpha^{[i,j]} = C_\alpha \cap [i,j]$ and $C_\alpha^{[i,j]}(c) = C_\alpha(c) \cap [i,j]$ to restrict the elements of the sets within the range $[i,j]$.

By Definition 1, $C_\alpha^{[1,c]}(c) = \{c'\} \cup C_\alpha^{[1,c']}(c')$ where $c' = \max C_\alpha^{[1,c]}(c)$. Consider a tree such that $C_\alpha \cup \{\perp\}$ is the set of nodes where $\perp$ is the root, and $\{(c', c) \mid c \in C_\alpha, c' = \max C_\alpha^{[1,c]}(c)\} \cup \{(\perp, c) \mid c \in C_\alpha, C_\alpha^{[1,c]}(c) = \emptyset\}$ the set of edges. This tree is called the *conflict tree* of α and it represents the relations of conflict points of α. Let $CT_\alpha(c)$ denote the set of children of node c and $CT_\alpha^{[i,j]}(c) = CT_\alpha(c) \cap [i,j]$. We define $\mathit{order}_\alpha(c)$ to be the depth of node c and $\mathit{maxc}_\alpha(c) = \max\{\mathit{order}_\alpha(c') \mid c' \in \{c\} \cup C_\alpha(c)\}$.

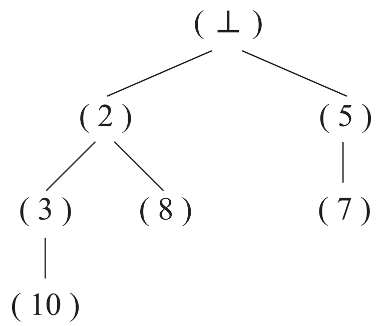

Fig. 1. The conflict tree of $\alpha = [0,1,1,2,3,4,3,1,2,1]$.

Fig. 1 illustrates the conflict tree for $\alpha = [0,1,1,2,3,4,3,1,2,1]$. Here $C_\alpha = \{2,3,5,7,8,10\}$, $C_\alpha(3) = \{2,10\}$, $CT_\alpha(2) = \{3,8\}$, $\mathit{order}_\alpha(2) = \mathit{order}_\alpha(5) = 1$, $\mathit{order}_\alpha(3) = \mathit{order}_\alpha(7) = \mathit{order}_\alpha(8) = 2$, $\mathit{order}_\alpha(10) = 3$, $\mathit{maxc}_\alpha(5) = \mathit{maxc}_\alpha(7) = \mathit{maxc}_\alpha(8) = 2$, $\mathit{maxc}_\alpha(2) = \mathit{maxc}_\alpha(3) = \mathit{maxc}_\alpha(10) = 3$, and so on.

Lemma 5 will be used to show the $O(n^{1.5})$ time complexity of our algorithm of Section 4.

Lemma 5. *For any $\alpha[1:n] \in \mathrm{A}$, $n \geq 1 + \sum_{c \in C_\alpha} \lfloor 2^{\mathit{order}_\alpha(c)-2} \rfloor$.*

Proof. Let $c_t \in C_\alpha$ with $t \geq 2$, $C_\alpha^{[1:c_t]}(c_t) = \{c_1, c_2, \ldots, c_{t-1}\}$ with $c_1 < c_2 < \cdots < c_t$. Let $m = \alpha[c_1] = \alpha[c_2] = \cdots = \alpha[c_t]$. By the definition of Γ, for any $1 < i \leq n$, $\alpha[i] \leq \alpha[i-1]+1$. Then, it follows from $(c_t - 1) - c_{t-1} \geq \alpha[c_t - 1] - \alpha[c_{t-1}]$ that $m + (c_t - 1) - c_{t-1} \geq \alpha[c_t - 1]$. Consequently, by Definition 1, we have $c_t \geq 2c_{t-1} - m$ from $\alpha[c_t - 1] \geq c_{t-1} - 1$. Hence, $c_t \geq 2c_{t-1} - m \geq 2^2 c_{t-2} - m(1+2) \geq \cdots \geq 2^{t-1}c_1 - m\sum_{i=0}^{t-2} 2^i = 2^{t-1}c_1 - m(2^{t-1}-1) = 2^{t-1}(c_1 - m) + m \geq 2^{t-1} + m$. It leads to $\alpha[c_t] - (\alpha[c_t - 1] + 1) \leq m - c_{t-1} \leq -2^{t-2}$. Since $\alpha[i] = 0$ and

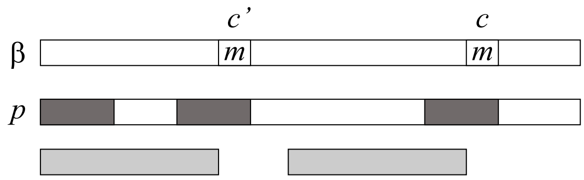

Fig. 2. Let $c, c' \in C_\beta$ and $\beta[c'] = \beta[c] = m$. Then, $c' \in C_\beta(c)$, $p[1 : m] = suf(p[1 : c'], m) = suf(p[1 : c], m)$, and $p[1 : c' - 1] = suf(p[1 : c - 1], c' - 1)$.

$1 \leq \alpha[i] \leq \alpha[i-1] + 1$ for any $1 < i \leq n$, $n - 1$ should be greater than the value subtracted over all conflict points. Therefore, the statement holds. □

The relation between conflict points of $\beta \in \mathrm{B}$ and $p \in \mathrm{P}_\beta$ is illustrated in Fig. 2.

Lemma 6 shows a necessary-and-sufficient condition for $\beta[1 : i]m$ to be a valid p-border array of some $p[1 : i+1] \in \mathcal{N}^*$, when $\beta[1 : i]$ is a valid p-border array.

Lemma 6. *Let $\beta[1 : i] \in \mathrm{B}$, $m \in \mathcal{N}$, and $p[1 : i+1] \in \mathcal{N}^*$. Then, $\beta[1 : i]m \in \mathrm{B}$ and $p[1 : i+1] \in \mathrm{P}_{\beta[1:i]m}$ if and only if*

$$\begin{aligned} & p[1 : i+1] \in \mathrm{P} \wedge p[1 : i] \in \mathrm{P}_{\beta[1:i]} \wedge \exists k, \beta^k[i] = m - 1 \wedge cut(p[i+1], m) = p[m] \\ \wedge & \big(C_{\beta[1:i]m}(i+1) \neq \emptyset \Rightarrow \big(p[m] = 0 \wedge \forall c \in C_{\beta[1:i]m}(i+1), p[i+1] \neq p[c] \\ & \quad \wedge \big(\exists c' \in C_{\beta[1:i]m}(i+1), p[c'] = 0 \Rightarrow m \leq p[i+1] < c'\big)\big)\big). \end{aligned}$$

Lemma 7 shows a yet stronger result, a necessary-and-sufficient condition for $\beta[1 : i]m$ to be a valid p-border array of length $i + 1$, when $\beta[1 : i]$ is a valid p-border array of length i.

Lemma 7. *Let $\beta[1 : i] \in \mathrm{B}$ and $m \in \mathcal{N}$. Then, $\beta[1 : i]m \in \mathrm{B}$ if and only if*

$$\begin{aligned} & \exists k, \beta^k[i] = m - 1 \\ \wedge & \big(C_{\beta[1:i]m}(i+1) \neq \emptyset \Rightarrow \big(\exists p[1 : i] \in \mathrm{P}_{\beta[1:i]} \ s.t. \ p[m] = 0 \\ & \wedge \big(\exists c' \in C_{\beta[1:i]m}(i+1), p[c'] = 0 \Rightarrow zeros(p[m : c' - 1]) \geq |C_{\beta[1:i]m}(i+1)|\big)\big)\big). \end{aligned}$$

Proofs of Lemmas 6 and 7 will be shown in a full version of this paper.

In the next section we design our algorithm to solve Problem 1 based on Lemmas 6 and 7.

4 Algorithm

This section presents our $O(n^{1.5})$-time $O(n)$-space algorithm to verify if a given integer array of length n is a valid p-border array for an unbounded alphabet.

4.1 Z-pattern Representation

Lemma 7 implies that, in order to check if $\beta[1:i]m \in \mathrm{B}$, it suffices for us to know if $p[i]$ is zero or non-zero for each i. Let $\star$ be a special symbol s.t. $\star \neq 0$. For any $p \in \mathrm{P}$ and $1 \leq i \leq |p|$, let $ptoz(p)[i] = 0$ if $p[i] = 0$, and $ptoz(p)[i] = \star$ otherwise. The sequence $ptoz(p) \in \{0, \star\}^*$ is called the *z-pattern* of p. For any $\beta \in \mathrm{B}$, let $\mathrm{Z}_\beta = \{ptoz(p) \mid p \in \mathrm{P}_\beta\}$.

The next lemma follows from Lemmas 3, 6, and 7.

Lemma 8. *Let $\beta \in \mathrm{B}$ and $z \in \{0, \star\}^*$. Then, $z \in \mathrm{Z}_\beta$ if and only if all of the following conditions hold for any $1 \leq i \leq |z|$:*

1. $i = 1 \Rightarrow z[i] = 0$.
2. $z[\beta[i]] = \star \Rightarrow z[i] = \star$.
3. $\exists c \in C_\beta, \exists k, i = \beta^k[c] \Rightarrow z[i] = 0$.
4. $\exists c \in C_\beta(i), z[c] = 0 \Rightarrow z[i] = \star$.
5. $i \in C_\beta \wedge zeros(z[\beta[i] : i-1]) < maxc_\beta(i) - 1 \Rightarrow z[i] = \star$.
6. $i \in C_\beta \wedge zeros(z[\beta[i] : i-1]) = order_\beta(i) - 1 \Rightarrow z[i] = 0$.

Let $E_\beta = \{i \mid \exists c \in C_\beta, \exists k, i = \beta^k[c]\}$. For any $z \in \mathrm{Z}_\beta$ and $i \in E_\beta$, $z[i]$ is always 0.

We check if a given integer array $y[1:n]$ is a valid p-border array in two steps.

Step 1: While scanning $y[1:n]$ from left to right, check whether $y[1:n] \in \mathrm{A}$ and whether each position i $(1 \leq i \leq n)$ of y satisfies Conditions 3 and 4 of Lemma 8. Also, we compute E_y, and $order_y(i)$ and $maxc_y(i)$ for each $i \in C_y$.

Step 2: For each $i = 1, 2, \ldots, n$, we determine the value of $z[i]$ so that the conditions of Lemma 8 hold.

If we can determine $z[i]$ for all $i = 1, 2, \ldots, n$ in Step 2, then the input array y is a p-border array of some $p \in \mathrm{P}$ such that $ptoz(p) = z$.

4.2 Pruning Techniques

Given an integer array y of length n, we inherently have to search $\{0, \star\}^n$ for a z-pattern $z \in \mathrm{Z}_y$. To achieve an efficient solution, we utilize the following pruning lemmas.

For any $\beta \in \mathrm{B}$ and $1 \leq i \leq |\beta|$, we write as $u[1:i] \in \mathrm{Z}^i_\beta$ if and only if $u[1:i] \in \{0, \star\}^*$ satisfies all the conditions of Lemma 8 for any j $(1 \leq j \leq i)$. For any $h > i$, let $z[h] = 0$ if $h \in E_\beta$, and leave it undefined otherwise. Clearly, for any $z \in \mathrm{Z}_\beta$ and $1 \leq i \leq |\beta|$, $z[1:i] \in Z^i_\beta$.

We can use the contraposition of the next lemma for pruning the search tree at each non-conflict point of y.

Lemma 9. *Let $\beta \in \mathrm{B}$ and $i \notin C_\beta$ $(2 \leq i \leq |\beta|)$. For any $u[1:i-1] \in \mathrm{Z}^{i-1}_\beta$, if $u[\beta[i]] = 0$ and there exists $z \in \mathrm{Z}_\beta$ s.t. $z[1:i] = u[1:i-1]\star$, then there exists $z' \in \mathrm{Z}_\beta$ s.t. $z'[1:i] = u[1:i-1]0$.*

Proof. For any $1 \le j \le |\beta|$, let $v[j] = 0$ if $j = i$, and $v[j] = z[j]$ otherwise. Now we show $v \in \mathrm{Z}_\beta$. $v[i]$ clearly holds all the conditions of Lemma 8. Since $v[j] = z[j]$ at any other points, $v[j]$ satisfies Conditions 1, 2, 3 and 4. Furthermore, for any $c \in C_\beta$, $v[c]$ holds Conditions 5 and 6, since $\mathit{zeros}(v[\beta[c] : c-1]) \ge \mathit{zeros}(z[\beta[c] : c-1])$ and $z[c]$ holds those conditions. □

Next, we discuss our pruning technique regarding conflict points of y. Let $\beta \in \mathrm{B}$. $c \in C_\beta$ is said to be an *active conflict point* of β, iff $E_\beta \cap (\{c\} \cup C_\beta(c)) = \emptyset$. Obviously, for any $z \in \mathrm{Z}_\beta$ and $c \in C_\beta$, $z[c] = 0$ if $E_\beta \cap \{c\} \ne \emptyset$ and $z[c] = \star$ if $E_\beta \cap C_\beta(c) \ne \emptyset$. Hence we never branch out at any inactive conflict point during the search for $z \in \mathrm{Z}_\beta$. Let AC_β be the set of active conflict points in β. Our pruning method for active conflict points is described in Lemma 10.

Lemma 10. *Let $\beta \in \mathrm{B}, i \in AC_\beta$ and $i \le r \le |\beta|$ with $|CT_\beta^{[1,r]}(i)| < 2$. For any $u[1 : i-1] \in \mathrm{Z}_\beta^{i-1}$, if $u[1 : i-1]0 \in \mathrm{Z}_\beta^{i}$ and there exists $z[1 : r] \in \mathrm{Z}_\beta^{r}$ s.t. $z[1 : i] = u[1 : i-1]\star$, then there exists $z'[1 : r] \in \mathrm{Z}_\beta^{r}$ s.t. $z'[1 : i] = u[1 : i-1]0$.*

In order to prove Lemma 10, particularly to ensure Conditions 5 and 6 of Lemma 8 hold, we will estimate the number of 0's within the range $[\beta[c], c-1]$ for each $c \in C_\beta$ that is obtained when the prefix of a z-pattern is $u[1 : i-1]0$. Here, for any $\alpha \in \mathrm{A}$ and $1 \le b \le |\alpha|$, let $F_\alpha(b) = \{b\} \cup \{b' \mid \exists k, b = \alpha^k[b']\}$ and $F_\alpha^{[i,j]}(b) = F_\alpha(b) \cap [i, j]$. Then, the number of 0's related to i within the range $[\beta[c], c-1]$ can be estimated by $|F_\beta^{[\beta[c],c-1]}(i)|$. The following lemmas show some properties of $F_\alpha(b)$ that are useful to prove Lemma 10 above.

Lemma 11. *Let $\alpha \in \mathrm{A}$. For any $1 \le b \le |\alpha|$ and $1 < i < |\alpha|$,*

$$|F_\alpha^{[\alpha[i+1],i]}(b)| - |F_\alpha^{[\alpha[i],i-1]}(b)| - \sum_{k=1}^{k'-1} |F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b)| = \begin{cases} 1 & \text{if } i \in F_\alpha(b) \text{ and } \\ & \alpha^{k'}[i] \notin F_\alpha(b), \\ 0 & \text{otherwise,} \end{cases}$$

where k' is the integer such that $\alpha^{k'}[i] = \alpha[i+1] - 1$.

Proof. Since $[\alpha[i+1]-1, i-1] = [\alpha^{k'}[i], \alpha^{k'-1}[i]-1] \cup [\alpha^{k'-1}[i], \alpha^{k'-2}[i]-1] \cup \cdots \cup [\alpha^1[i], i-1]$, $|F_\alpha^{[\alpha[i+1]-1,i-1]}(b)| = |F_\alpha^{[\alpha[i],i-1]}(b)| + \sum_{k=1}^{k'-1} |F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b)|$ (See Fig. 3). Then, the key is whether each of i and $\alpha[i+1]-1$ is in $F_\alpha(b)$ or not. Obviously, if $\alpha^{k'}[i] = \alpha[i+1]-1 \in F_\alpha(b)$, then $i \in F_\alpha(b)$. It leads to the statement. □

Lemma 11 implies that $|F_\alpha^{[\alpha[i],i-1]}(b)|$ is monotonically increasing for i.

Lemma 12. *Let $\alpha \in \mathrm{A}$ and $c', c \in C_\alpha$ with $c' \in C_\alpha^{[1,c]}(c)$. For any $1 \le b < c'$,*

$$|F_\alpha^{[m,c-1]}(b)| \ge |F_\alpha^{[\alpha[c-1],c-2]}(b)| + \sum_{k=1}^{k'-1} |F_\alpha^{[\alpha^{k+1}[c-1],\alpha^k[c-1]-1]}(b)| + 1,$$

where $m = \alpha[c'] = \alpha[c]$ and k' is the integer such that $\alpha^{k'}[c-1] = c'-1$.

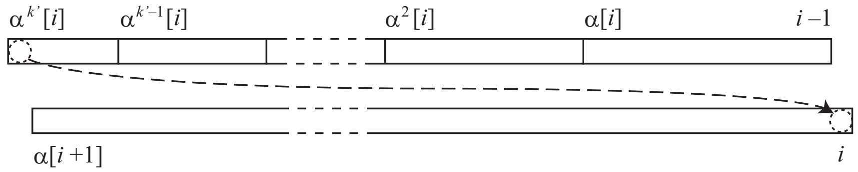

Fig. 3. Illustration for Lemma 11. If $\alpha^{k'}[i] = \alpha[i+1]-1 \in F_\alpha(b)$, then $i \in F_\alpha(b)$.

Proof. In a similar way to the proof of Lemma 11, we have $|F_\alpha^{[m,c-2]}(b)| = |F_\alpha^{[\alpha[c-1],c-2]}(b)| + \sum_{k=1}^{k'-1}|F_\alpha^{[\alpha^{k+1}[c-1],\alpha^k[c-1]-1]}(b)| + |F_\alpha^{[m,c'-2]}(b)|$. Since $c-1 \notin F_\alpha(b) \Rightarrow \alpha^{k'}[c-1] = c'-1 \notin F_\alpha(b)$,

$$|F_\alpha^{[m,c-1]}(b)| \geq |F_\alpha^{[\alpha[c-1],c-2]}(b)| + \sum_{k=1}^{k'-1}|F_\alpha^{[\alpha^{k+1}[c-1],\alpha^k[c-1]-1]}(b)| + |F_\alpha^{[m,c'-1]}(b)|.$$

Also, $|F_\alpha^{[m,c'-1]}(b)| \geq 1$ follows from Lemma 11. Hence, the lemma holds. □

Lemma 13. *For any* $\alpha \in \mathrm{A}, 1 \leq b < b' \leq |\alpha|$ *and* $1 \leq i < |\alpha|$, $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq |F_\alpha^{[\alpha[i+1],i]}(b')|$.

Proof. We will prove the lemma by induction on i. First, for any $1 \leq i < b$, it is clear that $|F_\alpha^{[\alpha[i+1],i]}(b)| = |F_\alpha^{[\alpha[i+1],i]}(b')| = 0$. Second, for any $b \leq i < b'$, it follows from Lemma 11 that $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq 1$. Then, $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq 1 > 0 = |F_\alpha^{[\alpha[i+1],i]}(b')|$. Finally, when $b' \leq i < |\alpha|$, let k' be the integer such that $\alpha^{k'}[i] = \alpha[i+1]-1$. (I) When $i \notin F_\alpha(b')$ or $\alpha^{k'}[i] = \alpha[i+1]-1 \in F_\alpha(b')$. It follows from Lemma 11 that $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq |F_\alpha^{[\alpha[i],i-1]}(b)| + \sum_{k=1}^{k'-1}|F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b)|$ and $|F_\alpha^{[\alpha[i+1],i]}(b')| = |F_\alpha^{[\alpha[i],i-1]}(b')| + \sum_{k=1}^{k'-1}|F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b')|$. By the induction hypothesis, we have $|F_\alpha^{[\alpha[i],i-1]}(b)| \geq |F_\alpha^{[\alpha[i],i-1]}(b')|$ and $|F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b)| \geq |F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b')|$ for any $1 \leq k \leq k'-1$. Hence, $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq |F_\alpha^{[\alpha[i+1],i]}(b')|$. (II) When $i \in F_\alpha(b')$ and $\alpha^{k'}[i] = \alpha[i+1]-1 \notin F_\alpha(b')$. There always exists $b' \in \{i, \alpha^1[i], \ldots, \alpha^{k'-1}[i]\}$, and therefore $|F_\alpha^{[\alpha[b'],b'-1]}(b)| \geq 1 > 0 = |F_\alpha^{[\alpha[b'],b'-1]}(b')|$. Then, $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq |F_\alpha^{[\alpha[i],i-1]}(b)| + \sum_{k=1}^{k'-1}|F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b)| \geq 1 + |F_\alpha^{[\alpha[i],i-1]}(b')| + \sum_{k=1}^{k'-1}|F_\alpha^{[\alpha^{k+1}[i],\alpha^k[i]-1]}(b')| = |F_\alpha^{[\alpha[i+1],i]}(b')|$. Hence, $|F_\alpha^{[\alpha[i+1],i]}(b)| \geq |F_\alpha^{[\alpha[i+1],i]}(b')|$. □

In a similar way, we have the next lemma.

Lemma 14. *Let* $\alpha \in \mathrm{A}$ *and* $c \in C_\alpha$ *with* $CT_\alpha(c) = \{c'\}$. *For any* $1 \leq i < |\alpha|$, $|F_\alpha^{[\alpha[i+1],i]}(c)| \geq \sum_{g \in G}|F_\alpha^{[\alpha[i+1],i]}(g)|$, *where* $G = (C_\alpha^{[c,|\alpha|]}(c) - c')$.

Now, we are ready to prove Lemma 10. We will use Lemmas 13 and 14.

Proof. Let $G = \{g \mid g \in C_\beta^{[i,r]}(i), z[g] = 0\}$. Let v be the sequence s.t. for each $1 \le j \le r$, $v[j] = 0$ if $j \in F_\beta(i)$, $v[j] = \star$ if there is $g \in G$ s.t. $j \in F_\beta(g)$, and $v[j] = z[j]$ otherwise.

Now we show $v \in \mathrm{Z}_\beta$. By the definition of v and $u[1 : i-1]0 \in \mathrm{Z}_\beta^i$, it is clear that $v[j]$ holds Conditions 1, 2, 3 and 4 of Lemma 8 for any $1 \le j \le r$. Furthermore, $u[1 : i-1]\star \in \mathrm{Z}_\beta^i$ means that $zeros(v[\beta[i] : i-1]) \ge maxc_\beta(i) - 1$. Hence, $v[c]$ satisfies Conditions 5 and 6 for any $c \in C_\beta^{[1,r]}(i)$ since $zeros(v[\beta[c] : c-1]) \ge zeros(v[\beta[i] : i-1])$ and $maxc_\beta(i) - 1 \ge maxc_\beta(c) - 1$. Then, as the proof of Lemma 9, we have only to show $zeros(v[\beta[c] : c-1]) \ge zeros(z[\beta[c] : c-1])$ for any $c \in C_\beta$. This can be proven by showing $|F_\beta^{[\beta[c],c-1]}(i)| \ge \sum_{g \in G} |F_\beta^{[\beta[c],c-1]}(g)|$. Since it is clear in case where $G = \emptyset$, we consider the case where $G \ne \emptyset$. Let $c' = CT_\beta(i)$. Note that $|CT_\beta(i)| = 1$ by the assumption. (I) When $z[c'] = 0$. Since $z[1 : r]$ satisfies Condition 4 of Lemma 8, $G = \{c'\}$. It follows from Lemma 13 that $|F_\beta^{[\beta[c],c-1]}(i)| \ge |F_\beta^{[\beta[c],c-1]}(c')|$ for any $c \in C_\beta^{[1,r]}$. (II) When $z[c'] \ne 0$. It follows from Lemma 14 that $|F_\beta^{[\beta[c],c-1]}(i)| \ge \sum_{g \in G} |F_\beta^{[\beta[c],c-1]}(g)|$ for any $c \in C_\beta^{[1,r]}$. Therefore, the lemma holds. □

4.3 Complexity Analysis

Algorithm 1 shows our algorithm that solves Problem 1.

Theorem 1. *Algorithm 1 solves Problem 1 in $O(n^{1.5})$ time and $O(n)$ space for an unbounded alphabet.*

Proof. The correctness should be clear from the discussions in the previous subsections.

Let us estimate the time complexity of Algorithm 1 until the `CheckPBA` function is called at Line 24. As in the failure function construction algorithm, the while loop of Line 6 is executed at most n times. Moreover, for any $1 \le i \le n$, the values of $z[i]$, $prevc[i]$, and $order[i]$ are updated at most once. When i is a conflict point, Line 20 is executed at most $order_y(i) - 1$ times. Hence, it follows from Lemma 5 that the total number of times Line 20 is executed is $\sum_{c \in C_y} (order_y(c) - 1) \le 1 + \sum_{c \in C_y} \lfloor 2^{order_y(c)-2} \rfloor \le n$.

Next, we show the `CheckPBA` function takes in $O(n^{1.5})$ time for any input $\alpha \in \mathrm{A}$. Let $2 \le r_1 < r_2 < \cdots < r_x \le n$ be the positions for which we execute Line 6 or 10 when we first visit these positions. If such positions do not exist, `CheckPBA` returns "valid" in $O(n)$ time. Let us consider $x \ge 1$. For any $1 \le t \le x$, let $z_t[1 : r_t - 1]$ denote the z-pattern when we first visit r_t and let $l_t = \min\{c \mid c \in AC_\alpha^{[1,r_t-1]}, z_t[c] = 0\}$. If $x = 1$ and such l_1 does not exist, then `CheckPBA` returns "invalid" in $O(n)$ time. If $x > 1$, then there exists l_1 as we reach r_x. Furthermore, there exists l_t s.t. $l_t < r_1$ since otherwise we cannot get across r_1. Henceforth, we may assume $l_1 \le l_2 \le \cdots \le l_x$ exist. Note that by the definition of active conflict points, all elements of $F_\alpha(l_t) - \{l_t\}$ are not conflict points, and therefore for any $b \in F_\alpha(l_t)$, $z_t[b] = 0$.

Algorithm 1: Algorithm to verify p-border array

Input: an integer array $y[1:n]$
Output: whether y is a valid p-border array or not
/* $zeros[1:n]$: $zeros[i] = zeros(z[1:i])$. $zeros[0] = 0$ for convenience. */
/* $sign[1:n]$: $sign[i] = 1$ if $i \in E_y$, $sign[i] = -1$ if $(C_y^{[i,n]}(i) \cap E_y) \neq \emptyset$. */
/* $prevc[1:n]$: $prevc[i] = \max C_y^{[1,i]}(i)$, $prevc[i] = 0$ otherwise. */
if $y[1:2] \neq [0,1]$ **then return** invalid;
$sign[1:n] \leftarrow [1,0,..,0]$; $prevc[1:n] \leftarrow [0,..,0]$; $order[1:n] \leftarrow [0,..,0]$; $maxc[1:n] \leftarrow [0,..,0]$;
for $i = 3$ **to** n **do**
    if $y[i] = y[i-1]+1$ **then** continue;
    $b' \leftarrow y[i-1]$; $b \leftarrow y[b']$;
    while $b > 0$ & $y[i] \neq y[b'+1]$ & $y[i] \neq b+1$ **do**
        $b' \leftarrow b$; $b \leftarrow y[b']$;
    if $y[i] = y[b'+1]$ **then** /* i conflicts with $b'+1$ */
        $j \leftarrow y[i]$;
        while $sign[j] = 0$ & $order[j] = 0$ **do** /* $z[y^1[i]], z[y^2[i]], \ldots, z[0]$ must be 0 */
            $sign[j] \leftarrow 1$; $j \leftarrow y[j]$;
        if $sign[j] = -1$ **then return** invalid;
        if $sign[j] \neq 1$ **then**
            $sign[j] \leftarrow 1$; $j \leftarrow prevc[j]$;
            while $j > 0$ **do** /* $\forall j \in C_y^{[1,i]}(i), z[j]$ must be $\star$ */
                if $sign[j] = 1$ **then return** invalid;
                $sign[j] \leftarrow -1$; $j \leftarrow prevc[j]$;
        if $order[b'+1] = 0$ **then** $order[b'+1] \leftarrow 1$;
        $prevc[i] \leftarrow b'+1$; $order[i] \leftarrow order[b'+1]+1$; $maxc[i] \leftarrow order[b'+1]+1$; $j \leftarrow b'+1$;
        while $j > 0$ & $maxc[j] < order[b'+1]+1$ **do**
            $maxc[j] \leftarrow order[b'+1]+1$; $j \leftarrow prevc[j]$;
    else if $y[i] \neq b+1$ **then return** invalid;
$cnt[1:n] \leftarrow [-1,..,-1]$; $zeros[1] \leftarrow 1$;
return CheckPBA($2, n, y[1:n], zeros[1:n], sign[1:n], cnt[1:n], prevc[1:n], order[1:n], maxc[1:n]$);

Here, let $L_1 = \{c \mid c \in C_\alpha^{[l_1+1,r_1]}, l_1 < \max C_\alpha^{[1,c]}(c)\}$ and $L_t = \{c \mid c \in C_\alpha^{[r_{t-1}+1,r_t]}, l_t < \max C_\alpha^{[1,c]}(c)\}$ for any $1 < t \leq x$. Since $L_1, L_2, \ldots, L_x$ are pairwise disjoint, $|L| = \sum_{t=1}^{x}|L_t|$, where $L = \bigcup_{t=1}^{x} L_t$. It follows from Lemma 12 that $|F_\alpha^{[\alpha[r_t],r_t-1]}(l_t)| - |F_\alpha^{[\alpha[r_{t-1}],r_{t-1}-1]}(l_t)| \geq |L_t|$. In addition, for any $1 \leq t \leq x$, let $E_t^{in} = E_\alpha \cap ([\alpha[r_t], r_t-1] - [\alpha[r_{t-1}], r_{t-1}-1]\})$ and $E_t^{out} = E_\alpha \cap ([\alpha[r_{t-1}], r_{t-1}-1] - [\alpha[r_t], r_t-1]\})$, where $[\alpha[r_0], r_0-1] = \emptyset$. Since for any

Function `CheckPBA`($i, n, y[1:n], zeros[1:n], sign[1:n], cnt[1:n], prevc[1:n], order[1:n], maxc[1:n]$)

Result: whether y is a valid p-border array or not

if $i = n$ **then** **return** valid;
if $order[i] = 0$ **then** `/*` i `is not a conflict point */`
    $zeros[i] \leftarrow zeros[i-1] + zeros[y[i]] - zeros[y[i]-1]$;
    return `CheckPBA`($i+1, n, y[1:n], \ldots, maxc[1:n]$);
if $sign[i] = 1$ **then** `/*` $z[i]$ `must be 0 */`
    if $zeros[i-1] - zeros[y[i]-1] < maxc[i] - 1$ **then** **return** invalid;
    $zeros[i] \leftarrow zeros[i-1] + 1$;
    return `CheckPBA`($i+1, n, y[1:n], \ldots, maxc[1:n]$);
if $sign[i] = -1 \parallel zeros[i-1] - zeros[y[i]-1] < maxc[i] - 1$ **then** `/*` $z[i]$ `must be` $\star$ `*/`
    if $zeros[i-1] - zeros[y[i]-1] < order[i]$ **then** **return** invalid;
    $zeros[i] \leftarrow zeros[i-1]$;
    return `CheckPBA`($i+1, n, y[1:n], \ldots, maxc[1:n]$);
`/* from here` $sign[i] = 0$ `and` $zeros[i-1] - zeros[y[i]-1] \geq maxc[i] - 1$ `*/`
if $cnt[i] = -1$ **then** `/* first time arriving at` i `*/`
    $cnt[i]{+}{+}$; $cnt[prevc[i]]{+}{+}$
if $prevc[i] > 0$ & $sign[prevc[i]] = 1$ **then** `/*` $\exists c \in C_y^{[1,i]}(i), z[c] = 0$ `*/`
    $sign[i] \leftarrow 1$; $zeros[i] \leftarrow zeros[i-1]$;
    $ret \leftarrow$ `CheckPBA`($i+1, n, y[1:n], \ldots, maxc[1:n]$); $sign[i] \leftarrow 0$;
    return ret;
$sign[i] \leftarrow 1$; $zeros[i] \leftarrow zeros[i-1] + 1$;
$ret \leftarrow$ `CheckPBA`($i+1, n, y[1:n], \ldots, maxc[1:n]$); $sign[i] \leftarrow 0$;
if $ret =$ valid $\parallel cnt[i] < 2$ **then** **return** ret;
$zeros[i] \leftarrow zeros[i-1]$;
return `CheckPBA`($i+1, n, y[1:n], \ldots, maxc[1:n]$);

$1 < t \leq x$, $zeros(z_t[\alpha[r_{t-1}] : r_{t-1} - 1]) \geq zeros(z_{t-1}[\alpha[r_{t-1}] : r_{t-1} - 1]) + 1$,

$$
\begin{aligned}
& zeros(z_t[\alpha[r_t] : r_t - 1]) \\
\geq\ & zeros(z_t[\alpha[r_{t-1}] : r_{t-1} - 1]) + |E_t^{in}| - |E_t^{out}| \\
& + |F_\alpha^{[\alpha[r_t], r_t - 1]}(l_t)| - |F_\alpha^{[\alpha[r_{t-1}], r_{t-1} - 1]}(l_t)| \\
\geq\ & zeros(z_{t-1}[\alpha[r_{t-1}] : r_{t-1} - 1]) + 1 + |E_t^{in}| - |E_t^{out}| + |L_t|.
\end{aligned}
$$

By recursive procedures, we have $order_\alpha(r_x) \geq 1 + zeros(z_x[\alpha[r_x] : r_x - 1]) \geq zeros(z_1[\alpha[r_1] : r_1 - 1]) + x + \sum_{t=2}^{x} |E_t^{in}| - \sum_{t=2}^{x} |E_t^{out}| + \sum_{t=2}^{x} |L_t|$. Since $zeros(z_1[\alpha[r_1] : r_1 - 1]) \geq 1 + |E_1^{in}| + |L_1|$ and $\sum_{t=1}^{x} |E_t^{in}| - \sum_{t=2}^{x} |E_t^{out}| \geq 1$, then $order_\alpha(r_x) \geq 2 + x + |L|$.

Now, we evaluate the number of z-patterns we search for during the calls of `CheckPBA`. Let $C_2(t) = \{c \mid c \in C_\alpha^{[l_t, r_t]}, |CT_\alpha^{[l_t, r_t]}(c)| \geq 2\}$ for any $1 \leq t \leq x$ and $T' = \{1\} \cup \{t \mid 1 < t \leq x, l_{t-1} < l_t, |CT_\alpha^{[l_t, r_{t-1}]}(l_t)| = 0\}$. Let us assume $T' = \{t'_1, t'_2, \ldots, t'_{x'}\}$ with $1 = t'_1 < t'_2 < \cdots < t'_{x'} \leq x$. By Lemmas 9 and 10,

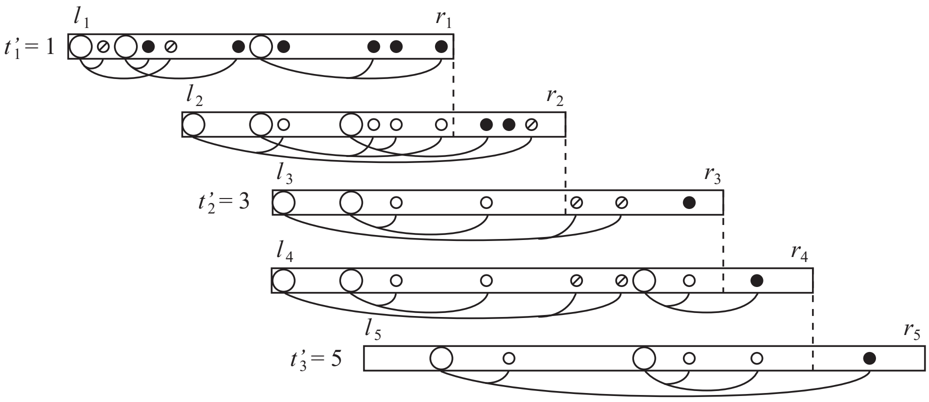

Fig. 4. Relation between L and C_2. A pair of a big circle and a small circle connected by an arc represents a parent-child relation in the conflict tree. $\bigcirc$ is a position in C. $\bullet$ or $\circ$ is a position in L. $\oslash$ is a position not in L.

the number of z-patterns searched for between $l_{t'_j}$ and $r_{t'_{j+1}-1}$ is at most $2^{|C'_2(t'_j)|}$ for any $1 \leq j \leq x'$, where $t'_{x'+1} - 1 = x$ and $C'_2(t'_j) = \bigcup_{t=t'_j}^{t'_{j+1}-1} C_2(t)$. Then, the total number of z-patterns is at most $\sum_{j=1}^{x'} 2^{|C'_2(t'_j)|}$. By Lemma 10, for any $1 \leq j < x'$, $l_{t'_j}$ must be in $C'_2(t'_j)$ and by the definition of T', $l_{t'_j}$ is only in $C'_2(t'_j)$. Hence, if $C_2 = \bigcup_{t=1}^{x} C_2(t)$, then $|C'_2(t'_j)| \leq |C_2| - (x' - 2)$, and therefore $\sum_{j=1}^{x'} 2^{|C'_2(t'_j)|} \leq 4x'2^{|C_2|-x'}$.

Finally, we consider the relation between L and C_2 (See Fig. 4). By the definition of L and C_2, for any $c \in (C_2 - \{l_1, l_2, \ldots, l_x\})$, $|CT_\alpha(c) \cap L| \geq 2$. In addition, by the definition of T', for any $c \in (C_2 \cap \{l_1, l_2, \ldots, l_x\} - \{l_{t'_1}, l_{t'_2}, \ldots, l_{t'_{x'}}\})$, $|CT_\alpha(c) \cap L| \geq 1$. Here, let $x'' = |\{l_1, l_2, \ldots, l_x\} - \{l_{t'_1}, l_{t'_2}, \ldots, l_{t'_{x'}}\}|$. Clearly, $x' + x'' \leq x$. For these reasons, $order_\alpha(r_x) \geq 2 + x + |L| \geq 2 + x + 2|C_2| - 2(x' + x'') + x'' \geq 2 + 2|C_2| - x'$. It follows from Lemma 5 that $n \geq 1 + \sum_{c \in C_\alpha} \lfloor 2^{order_\alpha(c)-2} \rfloor > 1 + \sum_{i=2}^{2+2|C_2|-x'} 2^{i-2} = 2^{2|C_2|-x'+1}$ and $\sqrt{n} > 2^{\frac{1+x'}{2}} 2^{|C_2|-x'} > x'2^{|C_2|-x'}$. Hence, the total time complexity is proportional to $n \sum_{j=1}^{x'} 2^{|C'_2(t'_j)|} \leq 4nx'2^{|C_2|-x'} < 4n\sqrt{n}$.

The space complexity is $O(n)$ as we use only a constant number of arrays of length n. □

5 Conclusions and Open Problems

We presented an $O(n^{1.5})$-time $O(n)$-space algorithm to verify if a given integer array y of length n is a valid p-border array for an unbounded alphabet. In case y is a valid p-border array, the proposed algorithm also computes a z-pattern

$z \in \{0, \star\}^*$ s.t. $z \in \mathrm{Z}_y$, and we remark that some sequence $p \in \mathrm{P}_y$ s.t. $ptoz(p) = z$ is then computable in linear time from z.

Open problems of interest are: (1) Can we solve the p-border array reverse problem for an unbounded alphabet in $o(n^{1.5})$ time? (2) Can we efficiently solve the p-border array reverse problem for a bounded alphabet? (3) Can we efficiently count p-border arrays of length n?

References

1. Baker, B.S.: Parameterized pattern matching: Algorithms and applications. Journal of Computer and System Sciences **52**(1) (1996) 28–42
2. Amir, A., Farach, M., Muthukrishnan, S.: Alphabet dependence in parameterized matching. Information Processing Letters **49**(3) (1994) 111–115
3. Kosaraju, S.: Faster algorithms for the construction of parameterized suffix trees. In: Proc. FOCS'95. (1995) 631–637
4. Hazay, C., Lewenstein, M., Sokol, D.: Approximate parameterized matching. ACM Transactions on Algorithms **3**(3) (2007) Article No. 29.
5. Apostolico, A., Erdös, P.L., Lewenstein, M.: Parameterized matching with mismatches. Journal of Discrete Algorithms **5**(1) (2007) 135–140
6. I, T., Deguchi, S., Bannai, H., Inenaga, S., Takeda, M.: Lightweight parameterized suffix array construction. In: Proc. IWOCA. (2009) 312–323
7. Idury, R.M., Schäffer, A.A.: Multiple matching of parameterized patterns. Theoretical Computer Science **154**(2) (1996) 203–224
8. Morris, J.H., Pratt, V.R.: A linear pattern-matching algorithm. Technical Report 40, University of California, Berkeley (1970)
9. I, T., Inenaga, S., Bannai, H., Takeda, M.: Counting parameterized border arrays for a binary alphabet. In: Proc. LATA'09. Volume 5457 of LNCS. (2009) 422–433
10. Franek, F., Gao, S., Lu, W., Ryan, P.J., Smyth, W.F., Sun, Y., Yang, L.: Verifying a border array in linear time. J. Comb. Math. and Comb. Comp. **42** (2002) 223–236
11. Duval, J.P., Lecroq, T., Lefevre, A.: Border array on bounded alphabet. Journal of Automata, Languages and Combinatorics **10**(1) (2005) 51–60
12. Duval, J.P., Lefebvre, A.: Words over an ordered alphabet and suffix permutations. Theoretical Informatics and Applications **36** (2002) 249–259
13. Bannai, H., Inenaga, S., Shinohara, A., Takeda, M.: Inferring strings from graphs and arrays. In: Proc. MFCS'03. Volume 2747 of LNCS. (2003) 208–217
14. Schürmann, K.B., Stoye, J.: Counting suffix arrays and strings. Theoretical Computer Science **395**(2-1) (2008) 220–234
15. Clément, J., Crochemore, M., Rindone, G.: Reverse engineering prefix tables. In: Proc. STACS'09. (2009) 289–300
16. Duval, J.P., Lecroq, T., Lefebvre, A.: Efficient validation and construction of border arrays and validation of string matching automata. RAIRO - Theoretical Informatics and Applications **43**(2) (2009) 281–297
17. Gawrychowski, P., Jez, A., Jez, L.: Validating the Knuth-Morris-Pratt failure function, fast and online. In: Proc. CSR'10. (2010) To appear.
18. Crochemore, M., Iliopoulos, C., Pissis, S., Tischler, G.: Cover array string reconstruction. In: Proc. CPM'10. (2010) To appear.
19. Moore, D., Smyth, W., Miller, D.: Counting distinct strings. Algorithmica **23**(1) (1999) 1–13