On-line Incremental Update Properties of the Subsequence Automaton

Alban Mancheron, Jean-Émile Symphor

On-line & Incremental Update Properties of the Subsequence Automaton
[version 1.1.2.9]

Alban Mancheron[†] and Jean-Émile Symphor[‡]

[‡] inria, Centre Lille – Nord-Europe, France.
alban.mancheron@inria.fr
[‡] Grimaag, Université des Antilles et de la Guyane, Martinique, France.
je.symphor@martinique.univ-ag.fr

Abstract. Many works deal with the subsequence matching problem using automata structures. It is to decide, given two sequences s and t, whether s is a subsequence of t. Automata like the Directed Acyclic Subsequence Graph (dasg) or the Subsequence Automaton (sa) accept all subsequences of a set of texts. We focus on this last structure and provide some useful results upon dynamically updates of the sa. Indeed, sequences are indexed as soon as they are processed, allowing to dynamically add or to remove sequences from the set of indexed texts. Moreover, the highlight of these properties also makes it possible to update this automaton whenever a sequence of the set is modified.

1 Introduction

In many research areas (such as network monitoring, molecular biology, data mining), extracting all subsequences from a set of sequences is one of the major issues on today. It allows to characterize some properties of the set of sequences. Before giving some known results, we need to introduce some usual definitions. A sequence s of symbols taken in a set Σ is called a string; the set Σ is then called the alphabet. A subsequence of a string s is a string t such that t can be obtained from s by deleting zero or more symbols.

1.1 Overview

In 1986, Hébrard and Crochemore proposed an algorithm for building the smallest deterministic and acyclic automaton that accepts all subsequences of a given sequence s [1]. Their algorithm processes the sequence s from right to left. They called this structure the Directed Acyclic Subsequence Graph (dasg). This way, Baeza-Yates extended the structure in 1991 for the case of multiple texts [2]. The dasg is now an acyclic and deterministic automaton that accepts all subsequences of any of the input texts. In the following, we denote the dasg of a set $\mathcal{S}$ by dasg($\mathcal{S}$). Troníček and Melichar introduced a left-to-right algorithm for constructing a dasg for one text and a quasi left-to-right algorithm

for building DASG($\mathcal{S}$) in 1998 [3] (see also [4]). A left-to-right algorithm allows to construct such an automaton without parsing the whole texts from $\mathcal{S}$a priori. We use the "quasi" qualifier because their algorithm processes the strings from left to right, but it requires to have a look straight forward in the sequences. However, in many applications (error detecting codes, weather sequences, patient treatment sequences) the sequences information grows incrementally with time. So, the existing automata construction algorithm are not suitable for handling this situation because the resulting automata obtained from previous sequences is no longer valid for the new sequences. In these situations, it is intolerably inefficient to restart the construction from scratch. In 2000, HOSHINO & al. [5] introduced a new structure close to the DASG, which accepts exactly the same language, which is still acyclic (with a slightly modification) and deterministic, called the Subsequence Automaton (SA); and they provide a left-to-right algorithm that constructs the structure (even if they do not prove the validity of their algorithm). What is new is that they provide a partially on-line and incremental algorithm for building the SA. The on-line attribute is due to the fact that given a set $\mathcal{S}$ of sequences, the SA obtained after the insertion of a new symbol at the end of the last sequence can be deduced from the previous one. In the same way, the incremental attribute is due to the fact that they also provide an algorithm which updates the SA in the case of new sequence insertion. Nevertheless, it can not be considered as a fully incremental and on-line algorithm.

1.2 Problem Statement

In a more general way, an incremental approach must consider the possibility of both insertion and deletion of complete sequences whatever the processing order of $\mathcal{S}$, whereas an on-line approach should allow both symbol insertion or deletion as it appears or disappears in any sequence from $\mathcal{S}$; and these two aspects have to preserve all the SA's properties. For instance, given a set $\mathcal{S}$ and a sequence s, SA($\mathcal{S} \cup \{s\}$) or SA($\mathcal{S} \setminus \{s\}$) (or a combination as SA($\mathcal{S} \setminus \{s\} \cup \{s'\}$), where s and s' differs only from one symbol α – insertion/deletion) can be deduced from SA($\mathcal{S}$) and s (and eventually α), thus each new sequence or new symbol is processed as soon as it is read (not necessary from left-to-right).

The main point of our contribution concerns the formalization of the incremental and on-line properties carried by the SA. This result allows in a first stage to prove the validity of the partially on-line and incremental algorithm provided in [5].

The subsequent is organized as follow: second section presents the definitions of the SA; in the third section are formally stated the on-line concepts related to the SA, and the incremental concepts are presented in the fourth section. In this last section, we consider the special case of the left-to-right insertion in any sequence.

2 Requirements

Before describe the subsequence automaton, we first recall some notations and definitions. First recall that Σ denotes a finite alphabet, thus we denote by ε the empty string (of length 0). Given a string $w \in \Sigma^*$, we denote by $Sub(w)$ the set of all subsequences of w and by $|w|$ the length of w. We denote by $w[i]$ $(1 \leq i \leq |w|)$ the i^{th} character of w and by $w[i..j]$ $(1 \leq i \leq j \leq |w|)$ the substring of w starting at position i and ending at position j, that is $w[i..j] = w[i] \cdots w[j]$. A subsequence of a string w is any string obtained by deleting zero or more symbols from w.

Let $\mathcal{S}$ denote a set of texts $\{s_1, \ldots, s_k\}$. Let n_i be the length of s_i for all $i \in [1; k]$. We say that t is a subsequence of $\mathcal{S}$ if and only if some $i \in [1; k]$ exists such that t is a subsequence of s_i. The SA($\mathcal{S}$) is a deterministic finite automaton which accepts all subsequences of s_i for all $i \in [1; k]$.

We use in this paper the standard notation of finite automata [6]. A finite automaton is a 5-tuple $(\mathcal{Q}, \Sigma, \delta, \mathcal{I}, \mathcal{F})$ where $\mathcal{Q}$ is a finite set of states, Σ an input alphabet, $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ is a transition function, $\mathcal{I} \subseteq \mathcal{Q}$ is the set of initial states and $\mathcal{F} \subseteq \mathcal{Q}$ is the set of final states.

In the following definitions, we keep the notations used in [7], which makes an overview of all the automata mentioned in the introduction. In the DASG presentation, is defined a set of **position points** of $\mathcal{S}$ named $\boldsymbol{Pos(\mathcal{S})}$. In the SA, this set is extended by adding the ∞ symbol in the position point definition. This is the reason why we use **extended position point** and $\boldsymbol{Pos'(\mathcal{S})}$ in the subsequent. The ∞ value corresponds to the last position after the last symbol of any string s_i, which is denoted by n_i in the definition of the position points.

Definition 1. *Given a set of strings $\mathcal{S} = \{s_1, \ldots, s_k\}$ from Σ^*, we define an extended position point of the set $\mathcal{S}$ as an ordered k-tuple $[p_1, \ldots, p_k]$, where $p_i \in [0; n_i] \cup \{\infty\}$ is a position in string s_i. If $p_i = 0$, then it denotes the empty string ε in front of the first position of s_i, if $p_i = \infty$, then it denotes the empty string behind the last position of s_i, otherwise it denotes the position of the ${p_i}^{th}$ symbol of s_i, for all $i \in [1; k]$.*

The particular position where $p_i = 0$ for all $i \in [1; k]$ is called the initial position point (denoted q_0^k in the subsequent). We also denote by $Pos'(\mathcal{S})$ the set of all extended position points of $\mathcal{S}$.

The set of extended position points is associated to the states of the SA($\mathcal{S}$), and the peculiar extended position point where all $p_i = \infty$ corresponds to a "sink state" (denoted q_∞^k). Now, let introduce the transition function used in this structure.

Definition 2. *Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ from Σ^*, an extended position point $[p_1, \cdots, p_k] \in Pos'(\mathcal{S})$ and $\alpha \in \Sigma$, we define the S-Reached Position Point (denoted $sRPP_{\mathcal{S}}([p_1, \cdots, p_k], \alpha)$) as $[p'_1, \cdots, p'_k]$, where $\forall\, i \in [1; k]$, $p'_i = \min\Big(\big\{j \mid j > p_i \wedge s_i[j] = \alpha\big\} \cup \big\{\infty\big\}\Big)$.*

Thus, we now can formally define the SA of a set of texts.

Definition 3. *Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ from Σ^*, we define the Subsequence Automaton for the strings $s_1, \cdots, s_k$ as the 5-tuple $\text{SA}(\mathcal{S}) = (\mathcal{Q}, \Sigma, \delta, \{q_0\}, \mathcal{F})$, where:*

$$\mathcal{Q} = Pos'(\mathcal{S}), \quad q_0 = q_0^k, \quad \delta = sRPP_{\mathcal{S}} \quad \text{and} \quad \mathcal{F} = Pos'(\mathcal{S}) \setminus \{q_\infty^k\}$$

Naturally, the $\text{SA}(\mathcal{S})$ accepts a string t if and only if $t \in \bigcup_{s \in \mathcal{S}} Sub(s)$. We illustrate in Fig. 1 an example of SA for three texts. We choose $s_1 = aba$ for the first string, $s_2 = aabb$ for the second one and $s_3 = aab$ for the third. We prune all the non final states but the sink one, since they could only lead to bottle-necks.

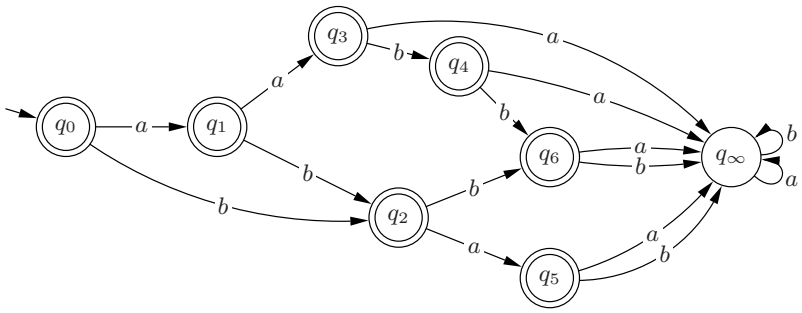

Fig. 1. $\text{SA}(\{aba, aabb, aab\})$, where $q_0 := q_0^3$, $q_1 := [1, 1, 1]$, $q_2 := [2, 3, 3]$, $q_3 := [3, 2, 2]$, $q_4 := [\infty, 3, 3]$, $q_5 := [3, \infty, \infty]$, $q6 := [\infty, 4, \infty]$ and $q_\infty := q_\infty^3$.

3 On-Line Update of the Subsequence Automaton

As a preamble, we need to make a trivial but important remark, due to the fact that we study the incremental and on-line update properties of the SA.

Remark 1. The definition of the SA still remains valid when considering multisets of strings instead of sets (two or more identical strings share exactly the same subsequences). Thus, when a string of $\mathcal{S}$ is updated, it could already exists in $\mathcal{S}$. So we don't make any distinction between set or multiset in the subsequent.

We also provide a useful lemma in order to simplify the notation used in following results.

Lemma 1 (Subsequence Automata isomorphism). *Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ and a permutation $\pi : [1; k] \to [1; k]$ such that $\mathcal{S}' = \{s'_1, \cdots, s'_k\}$ with $s'_i = s_{\pi(i)}$ for all $i \in [1; k]$, the $\text{SA}(\mathcal{S})$ and $\text{SA}(\mathcal{S}')$ are isomorphic relative to permutation π.*

Proof. Given an extended position point $p = [p_1, \cdots, p_k]$, we denote by $\pi(p)$ the tuple $[p_{\pi(1)}, \cdots, p_{\pi(k)}]$. This way, it is obvious that

$$Pos'(\mathcal{S}') = \Big\{ \pi(p) \;\Big|\; \exists\, p \in Pos'(\mathcal{S}) \Big\},$$
$$\text{and } sRPP_{\mathcal{S}'} = \Big\{ \big((\pi(p), \alpha) \mapsto \pi(q)\big) \;\Big|\; \exists\, \big((p, \alpha) \mapsto q\big) \in sRPP_{\mathcal{S}} \Big\}.$$

Thus, according to this lemma, we consider without loss of generality only updates of the last string of $\mathcal{S}$ in the subsequent. This makes the notations really more readable and understandable.

3.1 Symbol Insertion

Given two sets of strings $\mathcal{S}$ and $\mathcal{S}'$ of Σ^* such that $\mathcal{S} = \{s_1, \ldots, s_k\}$ and $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$, with $s_k = u\,v$ and $s'_k = u\,\alpha\,v$ ($u, v \in \Sigma^*$ and $\alpha \in \Sigma$), we want to insert the symbol α at the i^{th} position of s_k ($1 \leq i \leq n_k + 1$). Our objective is to deduce $\text{SA}(\mathcal{S}')$ from $\text{SA}(\mathcal{S})$, i and α. Let us consider the standard notation of the two resulted automata:

$$\text{SA}(\mathcal{S}) = (Pos'(\mathcal{S}), \Sigma, \delta_{\mathcal{S}}, \{q_0^k\}, Pos'(\mathcal{S}) \setminus \{q_\infty^k\})$$
$$\text{and} \quad \text{SA}(\mathcal{S}') = (Pos'(\mathcal{S}'), \Sigma, \delta_{\mathcal{S}'}, \{q_0^k\}, Pos'(\mathcal{S}') \setminus \{q_\infty^k\}),$$

where $\delta_{\mathcal{S}}$ and $\delta_{\mathcal{S}'}$ stand respectively for $sRPP_{\mathcal{S}}$ and $sRPP_{\mathcal{S}'}$.

So, our objective remains to define respectively each element of the 5-tuple $\text{SA}(\mathcal{S}')$ in accordance with the corresponding elements of the 5-tuple $\text{SA}(\mathcal{S})$ and the new position point and transitions relative to i and to the α symbol.

Remark 2. In these two automata, the initial state and the sink state are preserved because the set $\mathcal{S}'$ differs from $\mathcal{S}$ only by the symbol α in the last sequence, but they shared the same number of sequences.

As regards the states and the transitions which are modified or created as a result of the insertion of α, we proceed in three steps. Firstly, we define the new set of states of $\text{SA}(\mathcal{S}')$, then we took as a starting point the concept of non-solid transition presented in [5] (the authors consider the specific case where $i = n_k{+}1$) and we formalize this concept. Finally, we update the set of transitions.

So, particularly for the states, when inserting the symbol α at position i in the text s_k, we have to create all the extended position points having $n_k + 1$ as their k^{th} dimension. Notice that $n_k + 1$ is the length of s'_k.

Property 1. Given a set of k strings $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$ such that $|s'_k| = n_k + 1$ and the $\text{SA}(\mathcal{S})$:

$$Pos'(\mathcal{S}') = Pos'(\mathcal{S}) \uplus \Big\{ [p_1, \cdots, p_{k-1}, n_k + 1] \;\Big|\; \exists\, [p_1, \cdots, p_{k-1}, \infty] \in Pos'(\mathcal{S}) \Big\}.$$

In fact, introducing these new states corresponds to shift the k^{th} dimension for some states (where $p_k \geq i$), thus we need to apply these changes to the set of transitions $\delta_{\mathcal{S}}$.

Definition 4. *Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ from Σ^* and $i \in [1; n_k + 1]$, we define the set $\varphi_{\mathcal{S}}(i)$ such that*

$$\varphi_{\mathcal{S}}(i) = \Big\{ \big(([p_1, \cdots, p_{k-1}, p'_k], \alpha) \mapsto [r_1, \cdots, r_{k-1}, r'_k]\big) \;\Big|\; \exists \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \in \delta_{\mathcal{S}} \wedge p'_k = p_k + [\![p_k \geq i]\!] \wedge r'_k = r_k + [\![r_k \geq i]\!] \Big\},$$

where $[\![P]\!]$ stands for the Iverson*'s notation*[1].

Now, we need to define which transitions will be removed in $\text{SA}(\mathcal{S})$ in order to obtain $\text{SA}(\mathcal{S}')$. These are the one labeled by α that goes from a state where cursor p_k is before the position of the newly inserted symbol to a state where cursor r_k is after this position.

Definition 5. *Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ from Σ^*, a position $i \in [1; n_k + 1]$, and $\alpha \in \Sigma$, we define the set $\nabla_{\mathcal{S}}(\alpha, i)$ such that*

$$\nabla_{\mathcal{S}}(\alpha, i) = \Big\{ \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \in \varphi_{\mathcal{S}}(i) \;\Big|\; p_k < i \wedge r_k > i \Big\}.$$

Let $\big(([p_1, \ldots, p_k], \alpha) \mapsto [r_1, \ldots, r_k]\big)$ be a transition with respect to α in $\varphi_{\mathcal{S}}(i)$, and let $p_k < i$ and $r_k > i$. The inequality $r_k > i$ means that the character α appears after the i^{th} position of the text s_k (see Definition 2). Thus, the set $\nabla_{\mathcal{S}}(\alpha)$ represents all the transitions which should be modified by the insertion of the symbol α at position i (see Fig. 2).

Definition 6. *Given a set of strings $\mathcal{S} = \{s_1, \ldots, s_k\}$ from Σ^*, a position $i \in [1; n_k + 1]$ and $\alpha \in \Sigma$, we define the set $\triangle_{\mathcal{S}}(\alpha, i)$ such that*

$$\triangle_{\mathcal{S}}(\alpha, i) = \Big\{ \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_{k-1}, i]\big) \;\Big|\; \exists \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \in \nabla_{\mathcal{S}}(\alpha, i) \Big\},$$

The set $\triangle_{\mathcal{S}}(\alpha, i)$ represents the transitions which are present in $\text{SA}(\mathcal{S}')$ instead of those from $\nabla_{\mathcal{S}}(\alpha, i)$. They are the ingoing transitions of the new created states. In the same way, we define below their outgoing transitions. They corresponds (by Definition 2) to the s-Reachable Position Points starting at the extended position point associated to the new states for each symbol from Σ. If we consider such a transition, the $k-1$ first dimensions of the reachable position point can easily be deduced by considering any extended position point having the same $k-1$ starting positions as the created state. The last dimension of the reached position point can be deduced from any extended position having the k^{th} dimension starting at position $i-1$ in s_k (see Fig. 3).

[1] Recall that $[\![P]\!] = 1$ if P is true, 0 otherwise.

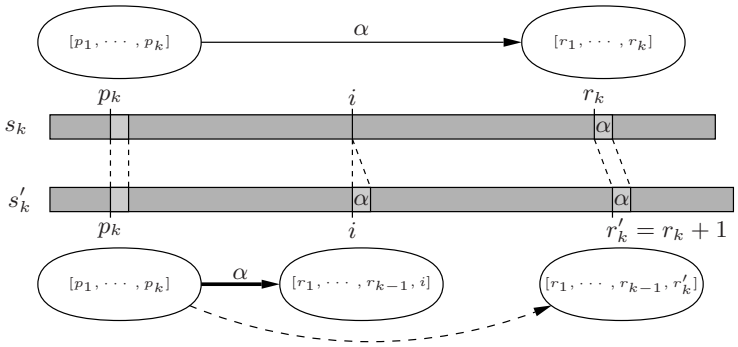

Fig. 2. Representation of the relationship between $\nabla_{\mathcal{S}}(\alpha, i)$ (the dashed transition) and $\triangle_{\mathcal{S}}(\alpha, i)$ (the bold transition) in $\textsc{sa}(\mathcal{S})$ and $\textsc{sa}(\mathcal{S}')$.

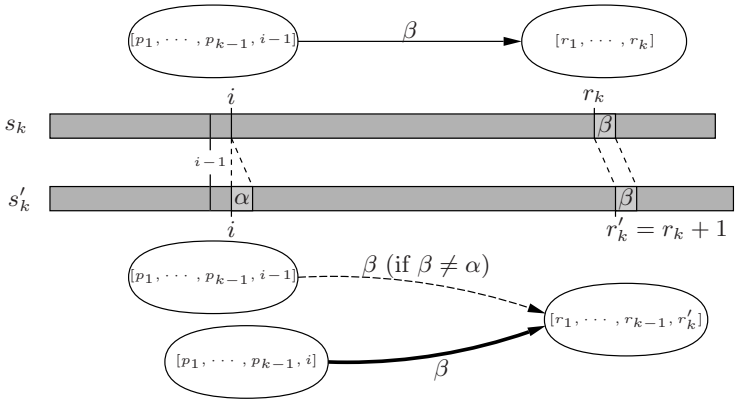

Fig. 3. Representation of the set $\Gamma_{\mathcal{S}}(i)$ (the bold transition) in $\textsc{sa}(\mathcal{S})$ and $\textsc{sa}(\mathcal{S}')$ (the dashed transition is in $\varphi_{\mathcal{S}}(i)$ and if $\beta = \alpha$, it is in $\nabla_{\mathcal{S}}(\alpha, i)$ too).

Definition 7. *Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ from Σ^* and $i \in [1; n_k + 1]$, we define the set $\Gamma_{\mathcal{S}}(i)$ such that*

$$\Gamma_{\mathcal{S}}(i) = \Big\{ \big(([p_1, \cdots, p_{k-1}, i], \beta) \mapsto [r_1, \cdots, r_k] \big) \;\Big|\; \exists \big(([p_1, \cdots, p_{k-1}, i-1], \beta) \mapsto [r_1, \cdots, r_k] \big) \in \varphi_{\mathcal{S}}(i) \Big\}.$$

Finally, using the four definitions above, we sum up the different results that allow to completely define the $\textsc{sa}(\mathcal{S}')$ from $\textsc{sa}(\mathcal{S})$.

Property 2. Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ from Σ^*, its subsequence automaton $\textsc{sa}(\mathcal{S})$ and $\alpha \in \Sigma$, let $\mathcal{S}'$ be $\mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$, with $s_k = u\,v$ and $s'_k = u\,\alpha\,v$ ($u, v \in \Sigma^*$ and $\alpha \in \Sigma$), the $\textsc{sa}(\mathcal{S}')$ is the 5-tuple defined by $(Pos'(\mathcal{S}'), \Sigma, \delta_{\mathcal{S}'}, \{q_0^k\}, Pos'(\mathcal{S}') \setminus \{q_\infty^k\})$, such that:

$$Pos'(\mathcal{S}') = Pos'(\mathcal{S}) \uplus \Big\{ [p_1, \cdots, p_{k-1}, n_k + 1] \;\Big|\; \exists\, [p_1, \cdots, p_{k-1}, \infty] \in Pos'(\mathcal{S}) \Big\}$$

and $\quad \delta_{\mathcal{S}'} = \varphi_{\mathcal{S}}(i) \setminus \nabla_{\mathcal{S}}(\alpha, i) \uplus \triangle_{\mathcal{S}}(\alpha, i) \uplus \Gamma_{\mathcal{S}}(i)$.

Remark 3. According to the Subsequence Automata isomorphism Lemma (p. 4), the Property 2 still remains applicable in order to deduce $\textsc{sa}(\mathcal{S} \setminus \{s_\ell\} \cup \{s'_\ell\})$ (for all $\ell \in [1;k]$) from $\textsc{sa}(\mathcal{S})$ (i.e. when inserting a character α to any $s_\ell \in \mathcal{S}$) by considering any permutation π such that $\pi(\ell) = k$.

3.2 Symbol Deletion

In order to deduce $\textsc{sa}(\mathcal{S})$ from $\textsc{sa}(\mathcal{S}')$, we proceed in a similar way as previously. Indeed, we just have shown that $\textsc{sa}(\mathcal{S})$ and $\textsc{sa}(\mathcal{S}')$ essentially differs in the set of extended position points and in the S-Reachable Position Points relationship.

In fact, it's rather easy[2] to understand that according to Property 2:

$$Pos'(\mathcal{S}) = Pos'(\mathcal{S}') \setminus \Big\{ [p_1, \cdots, p_{k-1}, n_k + 1] \;\Big|\; \exists\, [p_1, \cdots, p_{k-1}, \infty] \in Pos'(\mathcal{S}) \Big\},$$

and that $\delta_{\mathcal{S}}$ can be deduced from

$$\varphi_{\mathcal{S}}(i) = \delta_{\mathcal{S}'} \uplus \nabla_{\mathcal{S}}(\alpha, i) \setminus \triangle_{\mathcal{S}}(\alpha, i) \setminus \mathit{\Gamma}_{\mathcal{S}}(i).$$

So, we just have to express $Pos'(\mathcal{S})$ and $\varphi_{\mathcal{S}}(i)$ according to $\mathcal{S}'$, i and α instead of $\mathcal{S}$, i and α. The first hand is quiet trivial.

Property 3. Given a set of strings $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$ such that $|s'_k| = n_k + 1$ and the $\textsc{sa}(\mathcal{S}')$:

$$Pos'(\mathcal{S}) = Pos'(\mathcal{S}') \setminus \Big\{ [p_1, \cdots, p_{k-1}, n_k + 1] \in Pos'(\mathcal{S}') \Big\}.$$

The second hand is a little bit much more difficult since $\triangle_{\mathcal{S}}(\alpha, i)$ is defined from $\nabla_{\mathcal{S}}(\alpha, i)$, which is itself defined from $\varphi_{\mathcal{S}}(i)$ and α, and since $\mathit{\Gamma}_{\mathcal{S}}(i)$ is defined from $\varphi_{\mathcal{S}}(i)$. We can express the set $\nabla_{\mathcal{S}}(\alpha, i)$ from $\triangle_{\mathcal{S}}(\alpha, i)$ and $\mathit{\Gamma}_{\mathcal{S}}(i)$, whereas the sets $\triangle_{\mathcal{S}}(\alpha, i)$ and $\mathit{\Gamma}_{\mathcal{S}}(i)$ can be rewritten according to $\mathcal{S}'$, α and i instead of according to $\varphi_{\mathcal{S}}(i)$ and α.

The set $\nabla_{\mathcal{S}}(\alpha, i)$ corresponds to the transition that are not in $\delta_{\mathcal{S}'}$ anymore, so we need to build them again from $\triangle_{\mathcal{S}}(\alpha, i)$ and $\mathit{\Gamma}_{\mathcal{S}}(i)$ (see Fig. 4).

Property 4. Given a set of strings $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$ such that $s_k = u\,v$ and $s'_k = u\,\alpha\,v$ $(u, v \in \Sigma^*, \alpha \in \Sigma)$ and the $\textsc{sa}(\mathcal{S}')$, let $i = |u| + 1$, the set $\nabla_{\mathcal{S}}(\alpha, i)$ can be deduced from $\textsc{sa}(\mathcal{S}')$:

$$\begin{aligned} \nabla_{\mathcal{S}}(\alpha, i) = \Big\{ & \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k] \big) \;\Big| \\ & \exists \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_{k-1}, i] \big) \in \delta_{\mathcal{S}'} \wedge \\ & \exists \big(([r_1, \cdots, r_{k-1}, i], \alpha) \mapsto [v_1, \cdots, v_{k-1}, r_k] \big) \in \delta_{\mathcal{S}'} \Big\}, \end{aligned}$$

[2] Recall that unions are disjoint in Property 2 (by Definitions 5, 6 and 7).

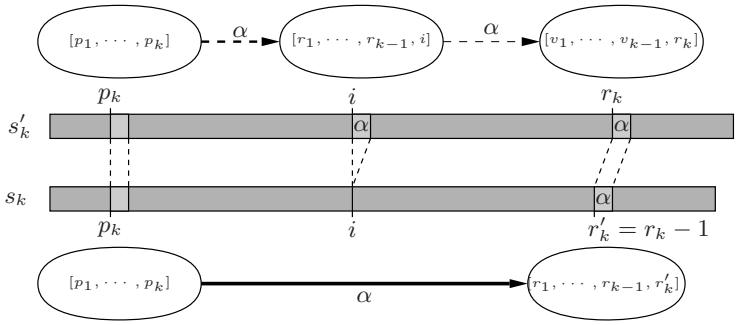

Fig. 4. Representation of the relationship between $\varphi_{\mathcal{S}}(i)$ and $\nabla_{\mathcal{S}}(\alpha, i)$ (the bold transition), $\triangle_{\mathcal{S}}(\alpha, i)$ (the thick dashed transition) and $\Gamma_{\mathcal{S}}(i)$ (the thin dashed transition) in $\text{SA}(\mathcal{S}')$ and $\text{SA}(\mathcal{S})$.

Now, let consider the set $\delta_{\mathcal{S}'} \setminus \triangle_{\mathcal{S}}(\alpha, i) \setminus \Gamma_{\mathcal{S}}(i)$. It corresponds literally to the transitions from $\delta_{\mathcal{S}'}$ such that the k^{th} dimension of the position point associated to their starting state or their ending state is different from i.

Property 5. Given a set of strings $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$ such that $s_k = u\,v$ and $s'_k = u\,\alpha\,v$ $(u, v \in \Sigma^*, \alpha \in \Sigma)$ and the $\text{SA}(\mathcal{S}')$, let $i = |u| + 1$, the set $\delta_{\mathcal{S}'} \setminus \triangle_{\mathcal{S}}(\alpha, i) \setminus \Gamma_{\mathcal{S}}(i)$ can be deduced from $\text{SA}(\mathcal{S}')$:

$$\delta_{\mathcal{S}'} \setminus \triangle_{\mathcal{S}}(\alpha, i) \setminus \Gamma_{\mathcal{S}}(i) = \Big\{ \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \in \delta_{\mathcal{S}'} \;\Big|\; p_k \neq i \vee r_k \neq i \Big\}.$$

From these last properties, the set $\varphi_{\mathcal{S}}(i)$ can be expressed from $\delta_{\mathcal{S}'}$ and i.

Property 6. Given a set of strings $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$ such that $s_k = u\,v$ and $s'_k = u\,\alpha\,v$ $(u, v \in \Sigma^*, \alpha \in \Sigma)$ and the $\text{SA}(\mathcal{S}')$, let $i = |u| + 1$, the set $\varphi_{\mathcal{S}}(i)$ can be deduced from $\text{SA}(\mathcal{S}')$:

$$\begin{aligned}
\varphi_{\mathcal{S}}(i) = & \Big\{ \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \in \delta_{\mathcal{S}'} \;\Big|\; p_k \neq i \vee r_k \neq i \Big\} \\
& \uplus \Big\{ \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \;\Big|\; \\
& \quad \exists \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_{k-1}, i]\big) \in \delta_{\mathcal{S}'} \wedge \\
& \quad \exists \big(([r_1, \cdots, r_{k-1}, i], \alpha) \mapsto [v_1, \cdots, v_{k-1}, r_k]\big) \in \delta_{\mathcal{S}'} \Big\},
\end{aligned}$$

It is quiet obvious that given $\varphi_{\mathcal{S}}(i)$ (for all $i \in [1; n_k + 1]$) we can build the set $\delta_{\mathcal{S}}$.

Property 7. Given a set of strings $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s'_k\}$ such that $s_k = u\,v$ and $s'_k = u\,\alpha\,v$ $(u, v \in \Sigma^*, \alpha \in \Sigma)$ and the $\text{SA}(\mathcal{S}')$, let $i = |u| + 1$, the set $\delta_{\mathcal{S}}$ can be deduced from $\varphi_{\mathcal{S}}(i)$, and so from $\text{SA}(\mathcal{S}')$:

$$\begin{aligned}
\delta_{\mathcal{S}} = & \Big\{ \big(([p_1, \cdots, p_{k-1}, p'_k], \alpha) \mapsto [r_1, \cdots, r_{k-1}, r'_k]\big) \;\Big|\; \\
& \quad \exists \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \cdots, r_k]\big) \in \varphi_{\mathcal{S}}(i) \\
& \quad \wedge\, p'_k = p_k - [\![p_k > i]\!] \wedge r'_k = r_k - [\![r_k > i]\!] \Big\},
\end{aligned}$$

Notice that the Subsequence Automaton isomorphism Lemma (p. 4) still applies in the context of symbol deletion (see Remark 3).

4 Incremental Update of the Subsequence Automaton

We previously gave results for insertion or deletion of symbols in existing sequences indexed in a SA. We now consider the case of insertion or deletion of a complete sequence in the set of indexed strings.

4.1 Sequence Insertion

The sequence insertion can be considered as the addition of the empty string into the SA and as the sequential insertion of symbols in this new string (as proposed in [5]). Since Property 2 (extended by Remark 3) provides a way for inserting a symbol into any indexed string at any position in a SA, we especially need to enable the insertion of the empty string. By taking into account the Subsequence Automaton isomorphism Lemma (p. 4), we will consider without loss of generality that this empty string is the last indexed sequence.

Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ and its subsequence automaton $\text{SA}(\mathcal{S}) = (Pos'(\mathcal{S}), \Sigma, \delta_{\mathcal{S}}, \{q_0^k\}, Pos'(\mathcal{S}) \setminus \{q_\infty^k\})$, we need to build $\text{SA}(\mathcal{S} \cup \{\varepsilon\}) = (Pos'(\mathcal{S} \cup \{\varepsilon\}), \Sigma, \delta_{\mathcal{S}\cup\{\varepsilon\}}, \{q_0^{k+1}\}, Pos'(\mathcal{S} \cup \{\varepsilon\}) \setminus \{q_\infty^{k+1}\})$. From the definitions of the set of extended position points and of the S-Reached Position Points (p. 3), since no symbol $\alpha \in \Sigma$ occurs in ε, we provide the following property:

Property 8. Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ and its $\text{SA}(\mathcal{S})$, let $\mathcal{S}' = \mathcal{S} \cup \{\varepsilon\}$, the sets $Pos'(\mathcal{S}')$ and $\delta_{\mathcal{S}'}$ can be respectively deduced from the sets $Pos'(\mathcal{S})$ and $\delta_{\mathcal{S}}$:

$$
\begin{aligned}
Pos'(\mathcal{S}') &= \Big\{ [p_1, \cdots, p_k, \infty] \;\Big|\; \exists\, [p_1, \cdots, p_k] \in Pos'(\mathcal{S}) \setminus \{q_0^k\} \Big\} \uplus \{q_0^{k+1}\}, \\
\text{and } \delta_{\mathcal{S}'} &= \Big\{ ([p_1, \cdots, p_k, \infty], \alpha) \mapsto [r_1, \cdots, r_k, \infty] \;\Big|\; \\
&\qquad \exists \big(([p_1, \cdots, p_k], \alpha) \mapsto [r_1, \ldots, r_k]\big) \in \delta_{\mathcal{S}} \wedge [p_1, \cdots, p_k] \neq q_0^k \Big\} \\
&\quad \uplus \Big\{ (q_0^{k+1}, \alpha) \mapsto [r_1, \cdots, r_k, \infty] \;\Big|\; \exists \big((q_0^k, \alpha) \mapsto [r_1, \ldots, r_k]\big) \in \delta_{\mathcal{S}} \Big\}.
\end{aligned}
$$

Thus, it only remains to add symbols in the last sequence. If we insert symbols from left to right in sequence s_k, the formulas of $\varphi_{\mathcal{S}}(i)$, $\nabla_{\mathcal{S}}(\alpha, i)$ and $\Gamma(i)$ can be simplified.

Property 9. Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$, its $\textsc{sa}(\mathcal{S})$ and a symbol $\alpha \in \Sigma$, let $\mathcal{S}' = \mathcal{S} \setminus \{s_k\} \cup \{s_k\,\alpha\}$:

$$\begin{aligned}
\varphi_{\mathcal{S}}(n_k+1) &= \delta_{\mathcal{S}},\\
\nabla_{\mathcal{S}}(\alpha, n_k+1) &= \Big\{\big(([p_1,\cdots,p_k],\alpha) \mapsto [r_1,\cdots,r_k]\big) \in \delta_{\mathcal{S}} \;\Big|\; p_k \neq \infty \wedge r_k = \infty\Big\},\\
\triangle_{\mathcal{S}}(\alpha, n_k+1) &= \Big\{\big(([p_1,\cdots,p_k],\alpha) \mapsto [r_1,\cdots,r_{k-1},n_k+1]\big) \;\Big|\\
&\qquad \exists\big(([p_1,\cdots,p_k],\alpha) \mapsto [r_1,\cdots,r_k]\big) \in \nabla_{\mathcal{S}}(\alpha, n_k+1)\Big\},\\
\Gamma_{\mathcal{S}}(n_k+1) &= \Big\{\big(([p_1,\cdots,p_{k-1},n_k+1],\alpha) \mapsto [r_1,\cdots,r_k]\big) \;\Big|\\
&\qquad \exists\,\big(([p_1,\cdots,p_{k-1},\infty],\alpha) \mapsto [r_1,\cdots,r_k]\big) \in \delta_{\mathcal{S}}\Big\}.
\end{aligned}$$

One can observe that the set $\nabla_{\mathcal{S}}(\alpha, n_k+1)$ represents the transition considered as *non-solid* in [5] and the set $\Gamma_{\mathcal{S}}(n_k+1)$ is sufficient to understand the main idea of their algorithm named *AppendCharacter*.

Finally, as for the previous results, according to the Subsequence Automaton isomorphism Lemma (p. 4), sequence insertion can be achieved at any position.

4.2 Sequence Deletion

Given a set $\mathcal{S}$, such that $\mathcal{S} = \{s_1, \cdots, s_k\}$, of k ($k \geq 1$) strings of Σ^*, we are interesting to deduce the $\textsc{sa}(\mathcal{S}')$ such that $\mathcal{S}' = \mathcal{S} \setminus \{s_k\}$ in accordance with $\textsc{sa}(\mathcal{S})$. We recall that :

$$\begin{aligned}
\textsc{sa}(\mathcal{S}) &= \Big(Pos'(\mathcal{S}), \Sigma, \delta_{\mathcal{S}}, \{q_0^k\}, Pos'(\mathcal{S}) \setminus \{q_\infty^k\}\Big)\\
\text{and}\quad \textsc{sa}(\mathcal{S}') &= \Big(Pos'(\mathcal{S}'), \Sigma, \delta_{\mathcal{S}'}, \{q_0^{k-1}\}, Pos'(\mathcal{S}') \setminus \{q_\infty^{k-1}\}\Big),
\end{aligned}$$

As previously, we have to build the set of extended position points $Pos'(\mathcal{S}')$ from $Pos'(\mathcal{S})$ and to update the set of transitions $\delta_{\mathcal{S}}$ from $\delta_{\mathcal{S}'}$. It is obvious that it could be achieved by a reversal process as for sequence insertion; a.k.a. by deleting successively all symbols of the sequence to be removed then by reducing the dimension of each state in the automaton. But formally, it can be achieved in a more interesting way. Indeed, removing the sequence s_k, implies the suppression of the value p_k (at the k^{th} position) in all element from the set $Pos'(\mathcal{S})$ and to apply this modification to the set of transitions. So we can simply deduce the property below.

Property 10. Given a set of strings $\mathcal{S} = \{s_1, \cdots, s_k\}$ and its $\textsc{sa}(\mathcal{S})$, let $\mathcal{S}' = \mathcal{S} \setminus \{s_k\}$, the sets $Pos'(\mathcal{S}')$ and $\delta_{\mathcal{S}'}$ can be respectively deduced from $Pos'(\mathcal{S})$ and $\delta_{\mathcal{S}}$:

$$\begin{aligned}
Pos'(\mathcal{S}') &= \Big\{[p_1,\cdots,p_{k-1}] \;\Big|\; \exists\,[p_1,\cdots,p_k] \in Pos'(\mathcal{S})\},\\
\text{and } \delta_{\mathcal{S}'} &= \Big\{([p_1,\cdots,p_{k-1}],\alpha) \mapsto [r_1,\cdots,r_{k-1}] \;\Big|\\
&\qquad \exists\,\big(([p_1,\cdots,p_k],\alpha) \mapsto [r_1,\ldots,r_k]\big) \in \delta_{\mathcal{S}}\}.
\end{aligned}$$

One more time, the Subsequence Automaton isomorphism Lemma (p. 4) applies and this last property can be generalized to the deletion of any sequence from $\mathcal{S}$.

5 Conclusion

We highlight and formalize the on-line and incremental properties of the subsequence automaton. Our contributions allow to cope with dynamic sets of sequences; that is, the removal (see Properties 3, 6, 7 and 10) as well as the addition (see Properties 2, 8 and 9) of any sequence or symbol at any position. In an extended version of this paper, we provide some algorithms on the basis of these theoretical results. Especially in the case of left-to-right insertion (we give the proof of the validity of the *AppendCharacter* algorithm provided in [5] as well as a generalization of it) and right-to-left deletion of symbols. We also consider the special cases of a "sliding texts window" (sequences queue) and of a "matching texts window" (sequences stack).

Moreover, in [7] we provide an automaton called the Constrained Subsequence Automaton with quorum q (denoted $\textsc{csa}_q$). This automaton accepts all the strings which are subsequences of at least q strings from a set of k texts $\mathcal{S}$ ($q \in [1; k]$). The states and transitions of this new automaton are the same as those of the Subsequence Automaton. The only difference remains from the set of final states. But practically, since the non final states are bottle-necks in this structure, they are not built. So all the properties provided in this paper are theoretically applicable to the $\textsc{csa}_q$. Unfortunately, in practice that is not possible to provide a fully incremental and on-line $\textsc{csa}_q$ when $q > 1$ without keeping all the non final states reachable from the initial state. Therefore, we aim to develop an *approximated* Constrained Subsequence Automaton with quorum q ($\widehat{\textsc{csa}}_q$) on the basis of these results, where the incremental and on-line update can be achieved without keeping the whole set of non final states reachable from the initial state.

References

1. Hébrard, J.J., Crochemore, M.: Calcul de la distance par les sous-mots. Informatique Théorique et Applications **20**(4) (1986) 441–456
2. Baeza-Yates, R.: Searching Subsequences. Theoretical Computer Science (tcs) **78**(2) (1991) 363–376
3. Troníček, Z., Melichar, B.: Directed Acyclic Subsequence Graph. In Holub, J., Šimánek, M., eds.: Proceedings of the Prague Stringology Club Workshop '98, Czech Technical University in Prague, Czech Republic (1998) 107–118
4. Crochemore, M., Melichar, B., Troníček, Z. : Directed Acyclic Subsequence Graph – Overview. Journal of Discrete Algorithms **1**(3–4) (2003) 255–280
5. Hoshino, H., Shinohara, A., Takeda, M., Arikawa, S.: Online Construction of Subsequence Automata for Multiple Texts. In: Proceedings of the 7th International Symposium on String Processing and Information Retrieval. (2000) 146–152

6. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Longman Publishing Co., Inc., Boston, usa (1990)
7. Mancheron, A., Symphor, J.-É.: Finding Frequent Subsequences in a Set of Texts. Submitted to the 16th Symposium on Foundation on Computer Theoretical Sciences (fcts), Budapest, Hungary (2007)