International Journal of Foundations of Computer Science

On-line construction of a small automaton for a finite set of words

Maxime Crochemore

King's College London, Strand
London WC2R 2LS, England, UK.
and Université Paris-Est, France.
Maxime.Crochemore@kcl.ac.uk

Laura Giambruno

Dipartimento di Matematica e Informatica, Università di Palermo,
Palermo, 90100, Italy.
lgiambr@math.unipa.it

Alessio Langiu

Dipartimento di Matematica e Informatica, Università di Palermo,
Palermo, 90100, Italy.
alangiu@math.unipa.it

In this paper we describe a "light" algorithm for the on-line construction of a small automaton recognising a finite set of words. The algorithm runs in linear time. We carried out good experimental results on real dictionaries, on biological sequences and on the sets of suffixes (resp. factors) of a set of words that shows how our automaton is near to the minimal one. For the suffixes of a text, we propose a modified construction that leads to an even smaller automaton.

We moreover construct linear algorithms for the insertion and deletion of a word in a finite set, directly from the constructed automaton.

Keywords: Finite set of words; deterministic automata; minimal automata; online construction.

1991 Mathematics Subject Classification: 68Q45, 68R15, 68W32 ,68W27

1. Introduction

The aim of this paper is to design a "light" algorithm that builds a small automaton accepting a finite set of words and that works on-line in linear time. The study of algorithms for the construction of automata recognising finite languages is interesting for parsing natural text and for motif detection (see [4]). It is used also in many software like the intensively used BLAST [2]. In particular it is important to study algorithms with good time and space complexities since the dictionaries used for natural languages can contain a large number of words.

It is in general easy to construct an automaton recognising a given list of words. Initially the list can be represented by a trie (see [5]) and then, using an algorithm for tree minimisation (see [1], [8]), we can minimise the trie to get the minimal automaton of the finite set of words of the list. But this solution requires a large memory space to store the temporary large data structure.

Another solution was drafted by Revuz in his thesis ([10]) where he proposed a pseudo-minimisation algorithm that builds from set of words in lexicographic inverse order an automaton smaller than the trie, but that is not necessarily minimal. Anyway the solution is not completely experimentally tested and remains unpublished.

Other solutions were proposed recently by several authors (cf. [15], [13], [14], chapter 2 of [9], [12], [7], [6]). For instance Watson in [15] presented a semi-incremental algorithm for constructing minimal acyclic deterministic automata and Sgarbas et al. in [12] proposed an efficient algorithm to insert a word in a minimal acyclic deterministic automata in order to obtain yet a minimal automaton, but not so efficient on building the automaton for a set of words. In [7] Daciuk et al. also proposed an algorithm that constructs a minimal automaton for an ordered set of strings, by adding new strings one by one and minimizing the resulting automaton.

Here we propose an intermediate solution, similar to that one of Revuz, that is to build a rather small automaton with a light algorithm processing the list of words on-line in linear time on the length of the list, where the length of a list is the sum of the lengths of the elements in the list. The aim is not to get the corresponding minimal automaton but just a small enough structure. However, the minimal automaton can be later obtained with Revuz' algorithm [11] that works in linear time on the size of the acyclic automaton.

The algorithm works on lists satisfying the following condition: words are in right-to-left lexicographic order. Such hypothesis on the list is not limitative since list update is standard. Moreover with the light algorithm the automaton can possibly be built on demand and our solution avoids building a temporary large trie.

The advantages of our algorithm are simplicity, on-line construction and the fact that resulting automaton seems to be really close to minimal.

In particular, in this paper we show the results of experiments done on real dictionaries, on biological sequences and on the sets of suffixes (resp. factors) of a set of words. For each set we consider the ratio between the number of states (resp. edges) of the constructed automaton and the number of states (resp. edges) of the minimal automaton associated with the set. Such ratios happen to be fairly small. For the suffixes of a text we even propose a modified construction that leads to an even smaller automaton. We moreover consider the two operations on dictionaries of insertion and deletion of a word. We give two linear algorithms in order to insert (resp. delete) a word in a set operating directly on the automaton.

In Section 2, after some standard definitions, we define the iterative construction of the automaton for a list of words. We bring some examples of the non minimality of this automaton. In Section 3 we describe the on-line algorithm that builds the automaton and that works in linear time on the length of the list. We moreover bring the results of experiments done on real dictionaries, on biological sequences and on suffixes on the sets of suffixes (resp. factors) of a set of words. In Section 4 we give the algorithms to insert and delete a word directly from the automaton of a finite set of words. In Section 5 we deal with sets that are suffixes of a given word. We give a modified version of our algorithm that leads to an even smaller automaton. Conclusions are in Section 6.

2. The algorithm for a finite set of words

For definitions on automata we refer to [3] and to [8].

Let A be a finite alphabet. Let x in A^*, then we denote by $|x|$ the length of x, by $x[j]$ for $0 \le j < |x|$ the letter of index j in x and by $x[j\,.\,.\,k] = x[j]\ldots x[k]$. For any finite set X of words we will denote by $|X|$ the cardinality of X and we will denote by $\|X\|$ the sum of lengths of the elements in X, $\|X\| = \sum_{x\in X}|x|$. Let u be a word in A^*, we denote by $S(u)$ the set of the proper suffixes of u together with u.

A *deterministic automaton* over A, $\mathcal{A} = (Q, i, T, \delta)$ consists of a finite set Q of *states*, of the initial state i, of a subset $T \subseteq Q$ of *final* states and of a *transition function* $\delta : Q \times A \longrightarrow Q$. For each p, q in Q, a in A such that $\delta(p, a) = q$, we call $(p, a, \delta(p, a))$ an *edge* of $\mathcal{A}$. An edge $e = (p, a, q)$ is also denoted $p \xrightarrow{a} q$. A *path* is a sequence of consecutive edges. A path is successful if its ending state is a final state. Given an automaton $\mathcal{A}$, we denote by $L(\mathcal{A})$ the language recognised by $\mathcal{A}$.

Let $X = (x_0, \ldots, x_m)$ be a list of words in A^* such that the list obtained reversing each word in X is sorted according to the lexicographic order. We will build an automaton recognising X with an algorithm that processes the list of words on-line. In order to do this we define inductively a sequence of $m + 1$ automata $\mathcal{A}_X^0, \ldots, \mathcal{A}_X^m$, such that, for each k, the automaton $\mathcal{A}_X^k$ recognises the language $\{x_0, \ldots, x_k\}$. In particular $\mathcal{A}_X^m$ will recognises X.

In the following we define $\mathcal{A}_X^0$ and then, for each $k \in \{1, \ldots, m\}$, we define the automaton $\mathcal{A}_X^k$ from the automaton $\mathcal{A}_X^{k-1}$. In these automata we will define a unique final state without any outgoing transition that we call q_{fin}. For each k, we consider the following functions over the set of states of $\mathcal{A}_X^k$ with values in N defined, for each state j in $\mathcal{A}_X^k$, as:

- Height: $H(j)$ is the maximal length of paths from j to a final state.
- Paths toward final states: for $j \neq q_{fin}$, $PF(j)$ is the number of paths starting at j and ending at final states and $PF(q_{fin}) = 1$.
- Indegree: $Deg^-(j)$ is the number of edges ending at j.
- Outdegree: $Deg^+(j)$ is the number of edges starting at j.

2.1. *Definition of $\mathcal{A}_X^0$*

Let $\mathcal{A}_X^0 = (Q_0, i_0, T_0, \delta_0)$ be the deterministic automaton having as states $Q_0 = \{0, \ldots, |x_0|\}$, initial state $i_0 = 0$, final state $T_0 = \{|x_0|\}$ and transitions defined, for each i in $Q_0 \setminus \{|x_0|\}$, by $\delta_0(i, x_0[i]) = i + 1$. We will denote by q_{fin} the final state

$|x_0|$. For each k in $\{0,\ldots,m\}$, q_{fin} will be the unique final state in $\mathcal{A}_X^k$ with no edge going out from it. It is easy to prove that $L(\mathcal{A}_X^0) = \{x_0\}$. In Figure 1 we can see $\mathcal{A}_X^0$ for $X = (\texttt{aaa,ba,aab})$.

Fig. 1. The automata $\mathcal{A}_X^0$ (left) and $\mathcal{A}_X^1$ (right) for $X = (\texttt{aaa,ba,aab})$. Since u, the prefix in common between aaa and ba, is the empty word and since s, the suffix in common between aaa and ba is a, the automaton $\mathcal{A}_X^1$ is obtained from $\mathcal{A}_X^0$ by adding the edge $(0, \texttt{b}, 2)$.

2.2. *Definition of $\mathcal{A}_X^k$ from $\mathcal{A}_X^{k-1}$*

Assume $\mathcal{A}_X^{k-1} = (Q_{k-1}, i_{k-1}, T_{k-1}, \delta_{k-1})$ has been built and let us define $\mathcal{A}_X^k = (Q_k, i_k, T_k, \delta_k)$. We define $i_k = \{0\}$.

Let u be the longest prefix in common between x_k and the elements $\{x_0,\ldots,x_{k-1}\}$. Let s be the longest suffix in common between x_k and x_{k-1}. If $|s| \geq |x_k| - |u|$ then we redefine s as $x_k[|u|+1\,..\,|x_k|-1]$. Let us consider p the end state of the path c in $\mathcal{A}_X^{k-1}$ starting at 0 with label u. Let q be the state along the path from 0 with label x_{k-1} for which the sub-path from q to q_{fin} has label s.

Indegree-Control The general idea of the construction of $\mathcal{A}_X^k$ from $\mathcal{A}_X^{k-1}$ would be to add a path from p to q. See Figure 1. Anyway in general we cannot do this since we would add others words other than x_k, as we can see in Figure 2. This depends on the fact that on the path c there are states r with $Deg^-(r) > 1$. Thus, before adding a path from p to q, we have to do a transformation of the automaton like in Figure 3.

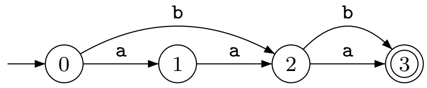

Fig. 2. Incorrect construction of $\mathcal{A}_X^2$ for $X = (\texttt{aaa,ba,aab})$: in this case $u = \texttt{aa}$ and $s = \varepsilon$, but, since $Deg^-(2) > 1$, adding the edge $(2, \texttt{b}, 3)$ leads to an automaton accepting $\{\texttt{aaa,ba,aab,bb}\}$.

More formally we consider separately the case in which there is a state on c with indegree greater than 1 and the other case.

I CASE: In c there is a state with indegree greater than 1.

Let us call r the first state with $Deg^-(r) > 1$. Let us decompose the path c as $c : 0 \xrightarrow{u_0} r_0 \xrightarrow{x[\ell]} r \xrightarrow{u_1} p$. We construct the automaton $\mathcal{B}_X^{k-1} = (Q'_{k-1}, 0, T'_{k-1}, \delta'_{k-1})$ in the following way. In order to construct δ'_{k-1}:

-we delete the edge $r_0 \xrightarrow{x[\ell]} r$,
-we construct a path from r_0 with label $x[\ell]u_1$, let us call p' its ending state,

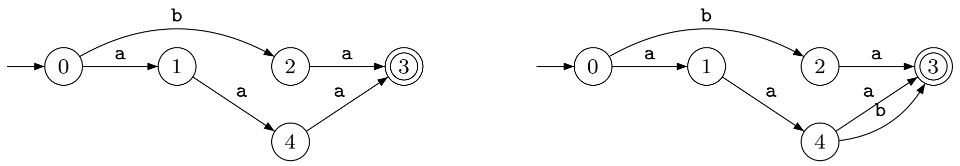

Fig. 3. Automata $\mathcal{B}_X^1$ (left) and $\mathcal{A}_X^2$ (right) for $X = (\mathtt{aaa}, \mathtt{ba}, \mathtt{aab})$. The automaton $\mathcal{B}_X^1$ is equivalent to $\mathcal{A}_X^1$ and it is obtained from $\mathcal{A}_X^1$ by doing a copy of the path from 0 to 4 with label $\mathtt{aa}$. The automaton $\mathcal{A}_X^2$ is obtained by adding the edge $(4, \mathtt{b}, 3)$

-we create for each edge going out from p with label a and ending at a state t, $p \xrightarrow{a} t$, the edge from p', $p' \xrightarrow{a} t$.
Moreover, if we duplicate a final state f then the duplicate will be a final state too. Following Lemma 2, f will be equal to q_{fin}. More formally we define $Q'_{k-1} = Q_{k-1} \cup \{|Q_{k-1}|, \ldots, |Q_{k-1}| + |u_1|\}$ and

$$\begin{cases} \delta'_{k-1}(i,a) = \delta_{k-1}(i,a), & \forall i \neq r_0, \forall a \in A; \\ \delta'_{k-1}(r_0, x_k[\ell]) = |Q_{k-1}|, & \\ \delta'_{k-1}(|Q_{k-1}| + i, x_k[\ell + i]) = |Q_{k-1}| + i + 1, & \forall i = 0, \ldots |u_1| - 1; \\ \delta'_{k-1}(|Q_{k-1}| + |u_1|, a) = \delta_{k-1}(p, a), & \forall a \in A. \end{cases}$$

We denote by p the state $|Q_{k-1}| + |u_1|$.

II CASE: the other case. We consider $\mathcal{B}_X^{k-1} = \mathcal{A}_X^{k-1}$.

We consider now the automaton $\mathcal{B}_X^{k-1}$. If x_k is the prefix of a word in $\{x_0, \ldots, x_{k-1}\}$ then we add p to the final states of $\mathcal{B}_X^{k-1}$, that is $T'_{k-1} = T_{k-1} \cup \{p\}$ and we define $\mathcal{A}_X^k = \mathcal{B}_X^{k-1}$.

Otherwise we proceed with the following control. We have the decomposition of x_k as $x_k = uws$, with $w \in A^+$.

Paths toward final states control As before, the general idea is to add a path from p to q with label w, but there are some other controls that are required. In Figure 4 we see another situation in which we cannot add a path from p to q otherwise we would add words not in X. In this case, it depends on the fact that the number of paths from q towards final states is greater than one, that is $PF[q] > 1$.

Thus, if $PF(q) > 1$ then we consider in the path d from q to q_{fin} with label s the first state q' such that $PF(q') = 1$, if it exists. In this case we call s_1 the label of the subpath of d from q to q' and let $s = s_1 s_2$. We call w the word ws_1, s the word s_2 and q the state q'. See Figure 4 for an example.

If there is no q' with $PF[q'] = 1$ in the path from q to q_{fin} with label s, then we define q as q_{fin} and w as ws. Otherwise we proceed with the Height control.

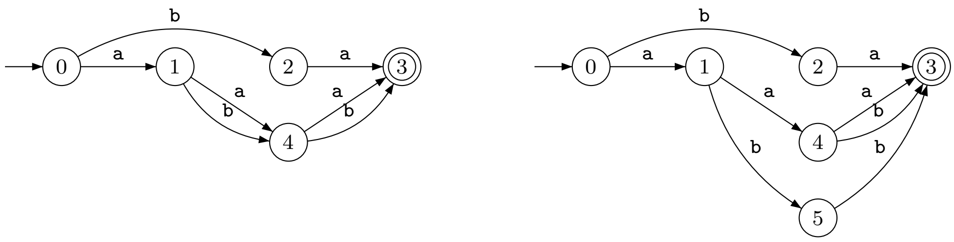

Fig. 4. Incorrect construction of $\mathcal{A}_X^3$ (left) for $X = (\texttt{aaa},\texttt{ba},\texttt{aab},\texttt{abb})$: adding the edge $(1, \texttt{b}, 4)$ to $\mathcal{A}_X^2$ leads to an automaton accepting $\{\texttt{aaa},\texttt{ba},\texttt{aab},\texttt{abb},\texttt{aba}\}$. Right construction of $\mathcal{A}_X^3$ (right): it is obtained by adding the path from 1 to 3 with label bb. In particular 3 is the first state q' in the path from 4 to 3 with $PF[q'] = 1$

Fig. 5. We have the automaton $\mathcal{A}_X^0$ (left) for $X = (\texttt{aba},\texttt{abbba})$ and the incorrect construction of $\mathcal{A}_X^1$ (right): adding the edge $(2, \texttt{b}, 1)$ would lead to an automaton accepting the infinite language $\{\texttt{aba},\texttt{a(bb)}^*\texttt{a}\}$.

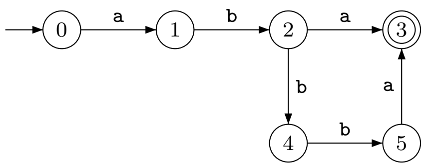

Fig. 6. We have the right construction of $\mathcal{A}_X^1$ for $X = (\texttt{aba},\texttt{abbba})$.

Height-Control If $H(p) \leq H(q)$ then in general we cannot add a path from p to q because if there is a path from q to p then we will have infinitely many words recognised, as we can see in the example in Figure 5. We have to do another transformation as in Figure 6.

If $H(p) \leq H(q)$ then we consider in the path d, from q to q_{fin} with label s, the first state q' such that $H(p) > H(q')$. We call s_1 the label of the subpath of d from q to q'. Let $s = s_1 s_2$. We call w the word ws_1, s the word s_2 and q the state q'. In Figure 6 we have an example of the construction.

If $H(p) > H(q)$ then we go further.

Another case that could create problem is when $p = q_{fin}$. In this case, if we add a path from p to q_{fin} with label w then we would add also infinitely many words to the language recognised by the automaton, as in the example in Figure 7. In Figure 7 it is also reported the right construction of the automaton, as explained in the following.

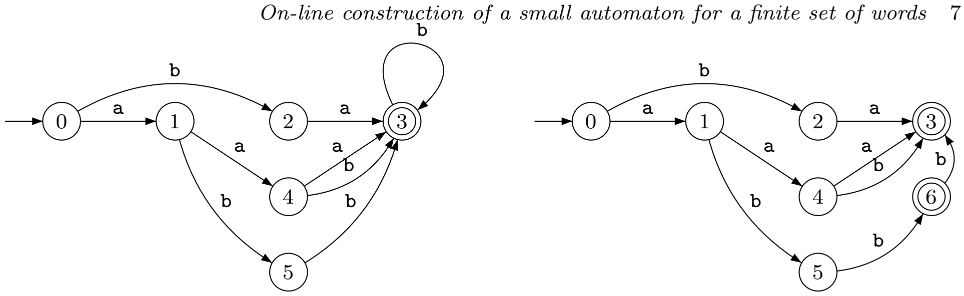

Fig. 7. Incorrect construction of $\mathcal{A}_X^4$ (left) and the right construction of $\mathcal{A}_X^4$ (right) for $X = (\texttt{aaa},\texttt{ba},\texttt{aab},\texttt{abb},\texttt{abbb})$.

When $p = q_{fin}$ we consider the following decomposition of c, the path from 0 with label u, $c : 0 \xrightarrow{u'} p' \xrightarrow{a} q_{fin}$. We delete the edge $p' \xrightarrow{a} q_{fin}$. Then we add an edge from p' to a new state p'' with label a and we add p'' to the set of final states. We call p the state p''. More formally we define $Q_k = Q'_{k-1} \cup \{|Q'_{k-1}|\}$ and

$$\begin{cases} \delta_k(i,a) = \delta'_{k-1}(i,a), \quad \forall i \neq p', \forall a \in A; \\ \delta_k(p',a) = |Q'_{k-1}|, \\ \delta_k(p',a) = \delta'_{k-1}(p',a), \, \forall b \in A, b \neq a; \end{cases}$$

We call p the state $|Q'_{k-1}|$.

Finally in all cases we add a path from p to q with label w, that is $Q_k = Q'_{k-1} \cup \{|Q'_{k-1}|+1, \ldots, |Q'_{k-1}|+|w|-3\}$ and

$$\begin{cases} \delta_k(i,a) = \delta'_{k-1}(i,a), & \forall i \neq p, \forall a \in A; \\ \delta_k(p,w[0]) = |Q'_{k-1}|, & \\ \delta_k(p,a) = \delta'_{k-1}(p,a), & \forall a \in A, a \neq w[0]; | \\ \delta_k(|Q'_{k-1}|+i, w[i+1]) = |Q'_{k-1}|+i+1, & \forall i = 0, \ldots |w|-3; \\ \delta_k(|Q'_{k-1}|+|w|-3, w[|w|-1]) = q, & \forall a \in A. \end{cases}$$

We have proved the following:

Theorem 1. *For each $k \in \{0,\ldots,m\}$, the language recognised by the automaton $\mathcal{A}_X^k$ is $L(\mathcal{A}_X^k) = \{x_0,\ldots,x_k\}$.*

In order to prove it we make use of the following lemma:

Lemma 2. *Let $k \in \{0,\ldots,m\}$. For each state i of $\mathcal{A}_X^k$ with $Deg^-(i) > 1$, there exists a unique path from i toward final states.*

Proof. We will prove the lemma by induction on k. For $k = 0$ it is easily true.

Let us suppose that it is true for $k-1$ and let us prove it for k. Let i be a state of $\mathcal{A}_X^k$ with $Deg^-(i) > 1$ then i is a state of $\mathcal{A}_X^{k-1}$ with $Deg^-(i) \geq 1$.

If $Deg^-(i) = 1$ in $\mathcal{A}_X^{k-1}$ then i is in $\mathcal{A}_X^k$ the ending state q of the path from p with label w relative to x_k and, by the PF control, one has that there is only one path from $i = q$ toward final states.

If $Deg^{-}(i) > 1$ in $\mathcal{A}_X^{k-1}$ then i is not contained in the path from 0 to p relative to x_k, by construction. So by the inductive hypothesis there is a unique path from i toward final states. □

By Lemma 2, it follows:

Corollary 3. *Let $k \in \{0, \ldots, m\}$. For each state i of $\mathcal{A}_X^k$ with $Deg^{-}(i) > 1$, there exists a unique path starting from i and ending at the final state q_{fin}.*

Proof of Theorem 1: We will prove the theorem by induction on k. For $k = 0$ it is easily true.

Let us suppose that it is true for $k-1$ and let us prove it for k. Let us prove that the automaton $\mathcal{B}_X^{k-1}$, obtained after the indegree control, recognises $\{x_0, \ldots, x_{k-1}\}$, that is $L(\mathcal{B}_X^{k-1}) = L(\mathcal{A}_X^{k-1})$. Let us suppose to be in CASE I otherwise it is trivial.

Trivially we have that $L(\mathcal{B}_X^{k-1}) \subseteq L(\mathcal{A}_X^{k-1})$. For the other inclusion, let d be a successful path in $\mathcal{A}_X^{k-1}$. If d does not contain the edge $r_0 \xrightarrow{x[\ell]} r$ then the path d will be also in $\mathcal{B}_X^{k-1}$. If d contains the edge $r_0 \xrightarrow{x[\ell]} r$ then d contains necessarily as subpath $r_0 \xrightarrow{x[\ell]} r \xrightarrow{u_1} p$, in fact, since $Deg^{-}(r_0) > 1$, by Corollary 3, there exists a unique path starting at r_0 and ending at q_{fin}. So there exists in $\mathcal{B}_X^{k-1}$ a successful path with the same label as d.

Let us prove now that the automaton $\mathcal{A}_X^k$ recognises $\{x_0, \ldots, x_k\}$. If x is the prefix of a word in $\{x_0, \ldots, x_{k-1}\}$ then we add p to the set of final states and, since the states in the path from 0 to p with label u have indegree equal to one, then we only add x_k to $L(\mathcal{A}_X^{k-1}) = \{x_0, \ldots, x_{k-1}\}$.

Otherwise, if $p = q_{fin}$ then we transform $\mathcal{B}_X^{k-1}$ in an automaton recognising the same language.

In all cases the automaton $\mathcal{A}_X^k$ is obtained from $\mathcal{B}_X^{k-1}$ by adding a path from p to q with label w, as defined before.

By the 'Indegree Control', there exists a unique path in $\mathcal{B}_X^{k-1}$ from 0 to p with label u and, by the 'paths toward final states control' there exists a unique path in $\mathcal{B}_X^{k-1}$ from q to q_{fin} with label s. Moreover, since $H(p) > H(q)$, there are no paths from q to p, otherwise there would be a path from q to q_{fin} longer than every path from p to q_{fin}.

Thus we only add to $L(\mathcal{A}_X^{k-1})$ the word $x = uws$, that is the thesis. □

2.3. *Non minimality of the automaton: example*

Given X a finite language, the automaton $\mathcal{A}_X$ is not necessarily minimal. This can follow, for example, from the not necessary indegree control done while building an automaton.

In the example in Figure 1 we see the construction of $\mathcal{A}_X^0$ and $\mathcal{A}_X^1$ for $X =$ (`aaa,ba,aab,bb`). In order to construct $\mathcal{A}_X^2$ we have to do the indegree control as

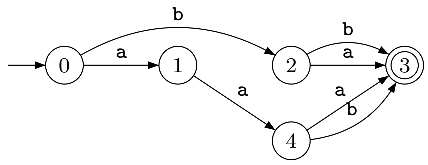

Fig. 8. Non minimal automaton $\mathcal{A}_X^3$ for $X = (\texttt{aaa,ba,aab,bb})$.

in Figure 3. In Figure 8 we have $\mathcal{A}_X^3$ that is not minimal since the states 2 and 4 are equivalent.

The non minimality follows here from the indegree control. In fact, in this case the indegree control would not be necessary since *bb* is also in X (see Figure 2). So the algorithm creates unnecessary states and the automaton $\mathcal{A}_X^3$ results to be non minimal.

3. Construction algorithm

Let $X = (x_0, \ldots, x_m)$ be a list of words in A^* ordered by right-to-left lexicographic order and let $\sum_{i=0,m} |x_i| = n$. Let us call $\mathcal{A}_X$ the automaton $\mathcal{A}_X^m$ recognising X. In order to build it on-line we have to go through all the automata $\mathcal{A}_X^k$, $0 \le k \le m$.

For the construction of $\mathcal{A}_X$ we consider a matrix of n lines and 3 columns where we will memorize the values of the three functions H, Deg^{-1} and PF for each state of the automaton. In the outline, when we write $\mathcal{A}$, we will consider the automaton $\mathcal{A}$ together with this matrix. The outline of the algorithm for computing $\mathcal{A}_X$ is the following:

```
Construction-A_X(X)
 1. (A, R) ← Construction-A_X^0 (X[0])
    ◁ denote by q_fin the final state of A, define PF[q_fin] = 1
 2. for k ← 1 to |X| − 1 do
 3.       (A, R) ← Add-word(A, X[k], X[k − 1], R)
 4. Return A
```

Let us consider now the function Construction-$\mathcal{A}_X$. In line 1 we have the function Construction-$\mathcal{A}_0$. Such a function computes the automaton $\mathcal{A}$ recognising $X[0]$. The automaton $\mathcal{A}$ is constructed using lists of adjacency. Its states are the integer $\{0, \ldots, |X[0]|\}$, 0 is the initial state and $|X[0]|$ is the final state. Moreover the function Construction-$\mathcal{A}_0$ returns a list R containing the sequence of states of $\mathcal{A}$ taken in the order of the construction.

In lines 2-3, for each k from 1 to $|X|$, we add to the automaton $\mathcal{A}$ the word $X[k]$ using the procedure Add-word below. Let us now see more in detail how the procedure Add-word works. It has as input an automaton $\mathcal{A}$ and two words x and y and it returns the automaton obtained from $\mathcal{A}$, by adding the word x, and R, the sequence of states along the path corresponding to the added word x. In line 1 it computes s the suffix in common x and y.

```
Add-word(A, x, y, R)
 1. compute s the suffix in common between x and y
 2. (A, j, p) ← Indegree-Control (A, x)
 3. if |x| = j then
 4.     add p to T
 5.     redefine PF for the states in the path from 0 with label x
 6.     define R as the list of states in the path from 0 with label x
 7. else
 8.     if |x| − |s| ≤ j then s ← s[j + 1 . . |s| − 1]
 9.     q ← R[|R| − |s|]
10.     (A, q, h) ← PF-Control(A, q, s)
11.     if PF[q] ≠ 1 then q_fin ← q
12.     else
13.         s ← s[h . . |s| − 1]
14.         (A, q, h) ← Height-Control(A, p, q, s)
15.     if p = q_fin then
16.         delete the last edge of the path c starting at 0 with label x[0 . . j − 1]
17.         add an edge from p_1 , ending state of c , to a new state p_2
18.         add p_2 to T
19.     (A, R) ← Add path(A, x[0 . . j − 1], x[j . . h − 1], q)
20.Return (A, R)
```

In line 2 it calls the INDEGREE-CONTROL function on $(\mathcal{A}, x)$ below.

```
Indegree-Control(A, x)
 1. p ← 0, j ← 0, InDegControl ← False
 2. while δ(p, x[j]) ≠ NIL and j ≠ |x|
 3.     p_1 ← p
 4.     p ← δ(p, x[j])
 5.     if InDegControl = False then
 6.         if Deg⁻[p] ≠ 1 then
 7.             create an edge from p_1 to a new state p_2 with label x[j]
 8.             define Deg⁻ for p_2 and for p
 9.             if p ∈ T then add p_2 to T
10.             InDegControl ← True
11.     else
12.         create an edge from p_2 to a new state q with label x[j]
13.         Deg⁻[q] ← 1
14.         p_2 ← q
15.         if p ∈ T then add p_2 to T
16.     j ← j + 1
17.if InDegControl = True then
18.     for each edge starting at p, with label a and ending state q
19.         create an edge from p_2 to q with label a
20.         Deg⁻[δ(p_2, a)] ← Deg⁻[δ(p_2, a)] + 1
21.     redefine the Height for all the new states created
22.     p ← p_2
23.Return (A, j, p)
```

Such a function reads the word x in $\mathcal{A}$ until it is possible. Let us call u the longest prefix of x that is the label of an accessible path in $\mathcal{A}$, let p be the ending state of this path. If, in such a path, there is an edge $r1 \xrightarrow{a} r$ such that r has indegree greater than 1 then the function creates a path from $r1$ labelled by the remaining part of u, let p_2 be its ending state. It redefines also the function Deg^- for the states in the new path.

In this case, for each edge starting at p with label a and ending at a state p', it creates an edge starting at p_2 with label a and ending at p'. It calls p the state p_2 and it redefines the height for the new states created. It moreover put on the set of final states of $\mathcal{A}$ the states created that are copies of final states of $\mathcal{A}$.

The INDEGREE-CONTROL function returns $\mathcal{A}$, j and p, where j is the length of the longest prefix of x which is the label of an accessible path in $\mathcal{A}$ and p is the ending state of this path.

Let us come back to ADD-WORD. In line 3 it controls if $x[0\,..\,j-1] = x$, that is if x is the prefix of an already seen word. In this case in lines 4-6 it puts p in the set of final states T, it redefines PF for the states on the path labelled x and it defines R as the list of states in the path from 0 with label x.

If $x[0\,..\,j-1] \neq x$, that is x is not the prefix of an already seen word, then we go to line 8. If $|x| - |s| \leq j$ then we redefine s. In line 9 we use R in order to find the state q such that there is a path from q to the final state q_{fin} with label s.

In line 10 the PF-CONTROL function is called. It takes as argument the automaton $\mathcal{A}$, q and s. The function reads from q the word s until either it finds a state q' with $PF[q'] = 1$ or it ends reading s. If s' is the label of the path from q to q' then it returns the length of such a path h. In line 11 if $PF[q]$ is greater than 1 then we define q as q_{fin}. Otherwise we go to line 13 where we redefine s as $s[h\,..\,|s|]$.

In line 14 we have a call to the HEIGHT-CONTROL function. It takes as argument the automaton $\mathcal{A}$, p, q and the word s. Such a function reads in $\mathcal{A}$, starting at q, the word s until it finds a state q' with $H[p] > H[q']$. If s' is the label of the path from q to q' then it returns the length of such a path h.

```
PF-Control (A, q, s)
h ← 0
while PF[q] ≠ 1 and h ≠ |s|
      q ← δ(q, s[h])
      h ← h + 1
Return (A, q, h)
```

```
Height-Control (A, p, q, s)
h ← 0
while H[p] ≤ H[q] and h ≠ |s|
      q ← δ(q, s[h])
      h ← h + 1
Return (A, q, h)
```

In line 15 it controls if $p = q_{fin}$. In such a case it does the transformation as written in lines 16, 17 and 18. In line 19 we call the function ADD-PATH on

$(\mathcal{A}, x[0\,..\,j-1], x[j\,..\,h-1], q)$.

The function ADD-PATH takes as argument $(\mathcal{A}, u, w, q)$ with u and w words and q state of $\mathcal{A}$. It returns the automaton $\mathcal{A}$ obtained by adding a path with label w from p, final state of the path in $\mathcal{A}$ from 0 with label u, to q. The function creates the path from p to q with label w and defines H, PF and Deg^- for the new states. It redefines H, PF and Deg^- for the states of the path from 0 with label u. Finally it puts all the states on the path from 0 to q_{fin} with label x in a list R. Then it returns the automaton $\mathcal{A}$ and the list R.

Time complexity

We define $\mathcal{A}_X^0$ using lists of adjacency. So we compute $\mathcal{A}_X^0$ with the associated matrix and R in $\mathcal{O}(|X[0]|)$. Let us analyze the time complexity of the other functions.

For each k, let us call u the longest prefix common to $X[k]$ and $\{X[0], \ldots, X[k-1]\}$. The INDEGREE-CONTROL function has time complexity $\mathcal{O}(|u|)$. Let us call x the word $X[k]$ and s the suffix in common between x and $X[k-1]$. The HEIGHT-CONTROL function works in $\mathcal{O}(h)$. The PF-CONTROL function works in $\mathcal{O}(h)$ also. Since $\mathcal{O}(h)$ are $\mathcal{O}(|s|)$ then the functions work in time $\mathcal{O}(|s|)$. The ADD PATH function works in time $\mathcal{O}(|x|)$.

Since the other instructions in ADD-WORD work in $\mathcal{O}(1)$ we get that the running time for executing ADD-WORD is $\mathcal{O}(|x|)$. And we get that the time complexity of CONSTRUCTION-$\mathcal{A}_X$ is $\mathcal{O}(\|X\|)$.

3.1. *Experimental Results*

In this section we show the results of the experiments. We show how our algorithm performs on some natural languages, namely the French language, the English language and the Italian language (see Table 1). For each set we consider the set cardinality ($|X|$), the sum of the lengths of the set words ($\|X\|$), the number of states and edges of the automaton built by our algorithm $\mathcal{A}_X$ and those ones of the minimal automaton $\mathcal{M}_X$. We also show the ratios between the number of states (resp. edges) of the constructed automaton and of the minimal automaton associated with the set.

The dictionary used in our test are the ones used by the WinEdt text editor program in order of make the spelling check on the typed text. They are freely available on the WinEdt web site at `http://www.winedt.org/Dict/`.

Firstly we investigated on some subset of increasing size of the French dictionary: some experiments were done using the first n words of the dictionary considered in its common order (see Table 2) and others using n random samples as set of words (see Table 5).

Furthermore, we did some test on a set of amino acid sequences. The protein sequences considered are the amino acid sequences of the C. elegans organism (see Table 3).

We considered some subset of different sizes, the set of their suffixes and the set

X	French dict.	English dict.	Italian dict.
$\|X\|$	136625	150918	277312
$\\|X\\|$	1303746	1395722	2733627
s_A = # states of A	136778	225410	304673
e_A = # edges of A	277131	386357	621220
s_M = # states of M	21950	66966	63696
e_M = # edges of M	51792	140662	157341
s_A/s_M ratio	6.23	3.37	4.78
e_A/e_M ratio	5.35	2.75	3.95

Table 1. Experiment results on the French dictionary, the English dictionary and the Italian Dictionary. Dictionaries came from WinEdt web site at `http://www.winedt.org/Dict/`

X	first 100	first 1000	first 10000	first 100000	all
$\|X\|$	100	1000	10000	100000	136625
$\\|X\\|$	172	3274	51341	841093	1303746
s_A = # states of A	14	450	7126	92525	136778
e_A = # edges of A	107	1443	17725	194412	277131
s_M = # states of M	14	240	2195	18079	21950
e_M = # edges of M	100	950	7108	45382	51792
s_A/s_M ratio	1	1.88	3.25	5.12	6.23
e_A/e_M ratio	1.07	1.52	2.49	4.28	5.35

Table 2. Experiment results on different lengths of the initial part of the French dictionary.

X	C.elegans 10KB	C.elegans 100KB	C.elegans 1MB
$\|X\|$	22	218	2055
$\\|X\\|$	9979	99783	991227
s_A = # states of A	9921	97548	925857
e_A = # edges of A	9941	97763	927911
s_M = # states of M	9921	97548	923869
e_M = # edges of M	9941	97763	925914
s_A/s_M ratio	1	1	1.002
e_A/e_M ratio	1	1	1.002

Table 3. Experimental results on different lengths of the initial part of the C.elegans protein sequence. C.Elegans proteomic data came from the EBI database available at `ftp://ftp.ebi.ac.uk/pub/databases/embl/genomes/Eukaryota/celegans/PROT`

of their factors, too (see Tables 4 and 5).

In general the experiments show a good behavior of our algorithm. The ratio

X	Ce	$Suff(Ce)$	$Fact(Ce)$
$\lvert X\rvert$	3	1000	120130
$\lVert X\rVert$	1000	175532	15608010
# states of A	996	39342	129145
# edges of A	997	40432	133610
# states of M	996	9842	17096
# edges of M	997	10826	18082
A/M # states	1	3.998	7.554
A/M # edges	1	3.735	7.389

Table 4. $Ce = \{$a C.elegans protein sequence of one thousand amino acids$\}$, $Suff(Ce)$ is the set of the suffixes of Ce and $Fact(Ce)$ is the set of the factors of Ce.

X	Fr	$Suff(Fr)$	$Fact(Fr)$
$\lvert X\rvert$	100	669	3245
$\lVert X\rVert$	940	4598	18489
# states of A	443	1210	2655
# edges of A	544	1973	4470
# states of M	443	614	591
# edges of M	531	1190	1163
A/M # states	1.023	1.971	4.492
A/M # edges	1.024	1.658	3.844

Table 5. $Fr = \{$100 samples randomly extracted from the French dictionary$\}$, $Suff(Fr)$ is the set of the suffixes of FR and $Fact(Fr)$ is the set of the factors of Fr.

between the number of states (edges) of our automaton and the minimal one, seem to be fairly small. The experiments are done on different sizes of the input set. It came out that the ratios grow in a poly logarithmic way on the automaton size.

It is interesting to see how our automaton for the molecular biological data is extremely close to the minimal one. For the suffixes and the factors of a set of words the experiments seem also good even if the ratios show that our automaton is relatively far from the minimal automaton.

For these kind of sets it is probably better to use algorithms that take into account the properties of such sets (see [5]).

4. Inserting and deleting a word: operation on automata

Let X be a finite set of words ordered by right-to-left lexicographic order. Let x a word not in X and let x_j be the greatest word in X that is smaller than x by right-to-left lexicographic. We can find in x_j in $\mathcal{O}(log(card(X)) + |x|)$ by using a dichotomic algorithm (see [5]). We can easily compute s, the greatest suffix in common between x and x_j, in time $\mathcal{O}(|x|)$.

By reading the word x_j in $\mathcal{A}_X$ we compute R the list of states of the path from 0 with label x_j in $\mathcal{O}(|x_j|)$. Then we apply the procedure ADD-WORD to $(\mathcal{A}_x, x, x_j, R)$ in $\mathcal{O}(|x|)$. Thus we have an algorithm to insert a word in a finite set X, by maintaining the construction of the algorithm, in $\mathcal{O}(log(|X|) + |x| + |x_M|)$, where $x_M = max_{x_k \in X}\{|x_k|\}$.

In order to delete a word x_k in X directly from the automaton $\mathcal{A}_X$, we have to appropriately delete some edges (and the relatives states) in the path from the initial state 0 with label x_k as in the Figure in 9.

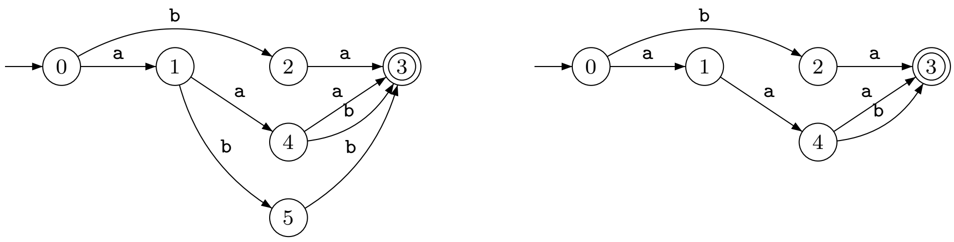

Fig. 9. The automaton $\mathcal{A}_X^3$ (left) for $X = (\texttt{aaa,ba,aab,abb})$ and the automaton $\mathcal{C}_3$ (rigth) for $X = (\texttt{aaa,ba,aab})$. It is obtained by $\mathcal{A}_X^3$ by deleting in the path α from 0 with label abb, the subpath that goes from 1 to 3 with label bb. Let us note that 1 is the last state in α with $PF(1) > 1$ and 3 is the first state in α with $Deg^-(3) < 1$.

Let α be a path and let i, j be two states of α. We say that i is before j, i.e. $i < j$, if there exists a path from i to j and it is a subpath of α. Let us note that, by Lemma 2, we cannot have in a path a state with indegree greater than one before a state with outdegree greater than one.

If, by reading x_k in $\mathcal{A}_X$, we arrive at an intermediate final state r then we consider the automaton $\mathcal{C}_k$ obtained from $\mathcal{A}_X$ by removing r by the set of final states of $\mathcal{A}_X$.

Otherwise, we consider the subpath β of the path α from 0 with label x_k starting at the last state i in α with $PF(i) > 1$ and ending at the first state j in α with $Deg^-(j) > 1$. If such a path does not exists then $\mathcal{A}_X$ is a single path and $X = \{x_k\}$.

We consider as $\mathcal{C}_k$ the automaton obtained from $\mathcal{A}_X$ by deleting the edges (and the relatives states) in β.

Proposition 4. *Let $k \in \{0, \ldots, m\}$. The automaton $\mathcal{C}_k$ recognizes $X \setminus \{x_k\}$.*

Proof. The case of x_k prefix of another word is trivial. Let so x_k be not a prefix of another word and let us prove that $X \setminus \{x_k\} \subseteq L(\mathcal{C}_k)$.

The word x_k is not recognized by $\mathcal{C}_k$ beacuse the automaton $\mathcal{A}_X$ is deterministic and we have removed in $\mathcal{A}_X$ at least one edge of the path with label x_k. Let

$x \in X \setminus \{x_k\}$ and let us consider the path in $\mathcal{A}_X$ from the initial state 0 with label x. Let us prove, by contradiction, that such a path do not visit edges and states removed from $\mathcal{A}_X$ and then it is contained also in $\mathcal{C}_k$.

Let us consider the ending state t of the path with label the longest prefix in common between x and x_k. The state t is a state with $PF(t) > 1$. The state t cannot be equal to q_{fin} since $Deg^+(t) \geq 1$ and there are not transitions going out from q_{fin}.

If the path from 0 with label x visits edges and states removed from $\mathcal{A}_X$ then t is either an intermediate state in the path β or it is a state in α after the states of β, otherwise in β there is a state with indegree greater than one.

Let t be an intermediate state in β. Since $PF(t) > 1$ then there is a contradiction by the construction of β. Let t be a state in α after the states of β. Since the last state of β has indegree greater than one, then there is a contradiction by Lemma 2.

Let us prove now that $L(\mathcal{C}_k) \subseteq X \setminus \{x_k\}$. Since we do not add edges and states then we have that $L(\mathcal{C}_k) \subseteq X$. Since x_k is not recognized by $\mathcal{C}_k$, then it follows the thesis. □

5. The algorithm for the set of suffixes of a given word

Let y in A^* and let us consider $S(y)$ sorted by decreasing order on the lengths of the elements in $S(y)$. For each $y \in A^*$, let us denote by $\mathcal{A}_y$ the automaton $\mathcal{A}_{S(y)}$ and by $\mathcal{M}_y$ the minimal automaton of $S(y)$. Given an automaton $\mathcal{A}$, let us denote by $\sharp\mathcal{A}$ the number of states of $\mathcal{A}$. For each y in A^*, in order to estimate the distance of $\mathcal{A}_y$ to its minimal automaton we consider the ratio $D(y) = \frac{\sharp\mathcal{A}_y}{\sharp\mathcal{M}_y}$.

We have done some experiments by generating all the words of a fixed length n. For each fixed length n we have considered D_n^{max} the greatest of $D(y)$ with y of length n.

n	D_n^{max}
10	1.83
15	2.41
20	3.04

In general the experimental results are good since D_n^{max} is not greater than 4 for words y with $|y| \leq 20$. Moreover the experiments done show that bad cases are linked with words that are powers of a short one with great exponent. So we thought that such words brought to automata far from being minimal (with great $D(y)$), or equivalently, that words with small entropy would have a great ratio $D(y)$.

Thus we have done experiments by generating 2000 words of a fixed length n with some constraints. For each of this experiment we have considered D_n, the greatest ratio among the $D(y)$. We report the results for different values of n in the following table. In the first column we have generated words such that either are not powers of the same word or that are powers of a word with an exponent less than a fixed number.

n	$exp < 3$	$exp < 2$	$exp < 1$
10	1.75	1.66	1.54
20	2.22	2.16	2.42
30	2.16	2.22	2.24
50	1.96	1.85	2.60
100	1.60	1.71	1.79

The experimental results are good in general even if they do not show clearly our conjecture. In the following we propose another approach.

5.1. *Modified construction*

Let y in A^* and $S(y) = [y_0 = y, \ldots, y_m]$ sorted by decreasing order on the lengths of the elements in $S(y)$. Let us denote by $\mathcal{A}_y^k$ the automaton $\mathcal{A}_{S(y)}^k$.

In case y_k is not a prefix of an already seen word, we consider the construction of the automaton $\mathcal{A}_y^k$ taking q in the path from 0 with label y and not in that one with label y_{k-1}. Let us note that in case of suffixes of a word we have that $y_k = uas$ with a in A and u, s defined as in Section 2. Moreover let us note that if there are two edges ending at p, state of $\mathcal{A}_y^k$, then they have the same label.

In this section we will propose a modification on the indegree control in order to avoid equivalent states as in the example in Figure 8. Before doing it we will note, with the following two propositions, that, in case of suffixes of a word, we do not have to execute the PF control and the Height control. In particular we prove that $PF(q) = 1$ and that $H(p) > H(q)$, with p and q as defined in Section 2.

Proposition 5. *Let y in A^* and y_k in $S(y)$ such that y_k is not a prefix of a word in $\{y_0, \ldots, y_{k-1}\}$. Then we have that $PF(q) = 1$.*

Proof. Let $y_k = uas$, with u and s as defined before. By contradiction, if $PF(q) > 1$ then, there exists $i < k$ such that $y_i = u_1bs_1$ and $|s| > |s_1|$. Since y_k is a suffix of y_i we have that $s = ts_1$, for some word $t \neq \varepsilon$. Since $y_k = uats_1$ is a suffix of $y_i = u_1bs_1$ then there exists $z \neq \varepsilon$ such that uz is a suffix of u_1.

Since $y_i = u_1bs_1$ then there exists y_h with $h < i$ such that u_1 is a prefix of y_h. Let $y_h = u_1cs_2$, then we have that $|s_2| > |s_1|$ since $|y_h| > |y_i|$. Since uz is a suffix of u_1 there exists a suffix y_l of y_h with $y_l = uzcs_2$. Since $|s_2| > |s_1|$ we get $|y_l| = |uzcs_2| > |uzbs_1| = |y_k|$. So uz is a prefix in common between y_k and y_l, $l < k$, that is a contradiction since u was the greatest prefix in common between y_k and the words in $\{y_0, \ldots, y_{k-1}\}$. □

Proposition 6. *Let y in A^* and y_k in $S(y)$ such that y_k is not a prefix of a word in $\{y_0, \ldots, y_{k-1}\}$. Then we have that $H(p) > H(q)$.*

Proof. Let $y_k = uas$ with u and s defined as before. Since p is co-accessible, there exists a word uz in $\{y_0, \ldots, y_{k-1}\}$. By Prop. 5, there exists only one path from q to q_{fin} whose label is s.
If $H(p) \leq H(q)$ then $|s| \geq |z|$ and so $|y_k| = |uas| > |uz|$ that is a contradiction since uz in in $\{y_0, \ldots, y_{k-1}\}$. □

Let us consider the construction of $\mathcal{A}_y^k$. We have the following proposition:

Proposition 7. *Let y in A^* and y_k in $S(y)$. Let $y_k = uz$, with u such that there exists a path starting at 0 with label u, let p its final state. If there exists a path from 0 to p with label v then, if $|v| < |u|$ then $vz \in \{y_{k+1}, \ldots, y_m\}$, otherwise $vz \in \{y_0, \ldots, y_{k-1}\}$.*

Proof. The state p is co-accessible and so let z' be the label of a path from p to a final state. Then uz' and vz' are in $\{y_0, \ldots, y_{k-1}\}$. If $|v| < |u|$ then v is a suffix of u and so vz is a suffix of uz and vz is in $\{y_{k+1}, \ldots, y_m\}$.

If $|v| \geq |u|$, then $|vz| \geq |uz|$. Thus uz is a suffix of vz and vz is in $\{y_0, \ldots, y_{k-1}\}$. □

Let us associate with each state p of $\mathcal{A}_y^{k-1}$ the list $L(p)$ of the states q such that there exists an edge from q to p. We construct such list iteratively adding each time an element to the tail of the list.

With each state q in $L(p)$ we associate $V(q)$ the set of words such that there exists a path from 0 to q. Let us denote by p_h the state $L(p)[h]$.

Proposition 8. *Let y in A^* and let p be a state of $\mathcal{A}_y^k$ with $Deg^-(p) > 1$. Let $i < j < |L(p)|$. Then, for each u in $V(p_i)$ and for each v in $V(p_j)$, we have that $|u| > |v|$.*

Proof. The list $L(p)$ is iteratively constructed adding each time an element to the tail of $L(p)$. Then, for each $i < j < |L(p)|$, and for u in $V(p_i)$ and v in $V(p_j)$, u is the label of a path added during the construction of $\mathcal{A}_y^l$, v is the label of a path added during the construction of $\mathcal{A}_y^r$, with $l < r$. Since p is co-accessible in $\mathcal{A}_y^l$, we have that uaz in $S(y)$ and so vaz in $S(y)$, for some word z and some a in A. Since v is constructed in $\mathcal{A}_y^r$ with $r > l$ we get that $|v| < |u|$. □

We propose a new construction of $\mathcal{A}_y^k$ with the definition of $L(p)$, for each state p, and a different indegree control. Let u be the longest prefix in common between y_k and $\{y_0, \ldots, y_{k-1}\}$ and p' the ending state of the path starting at 0 with label u.

The function reads the word y_k in $\mathcal{A}_y^{k-1}$ until it is possible. While the function reads y_k, the visiting state is called p and the state visited in the step before is called p_1. In particular we have that p_1 is in $L(p)$.

If the function finds a state p with Indegree greater than 1 and if p_1 is not equal to $L(p)[0]$ then, if a is the label of the edge from p_1 to p,

-it deletes all the edges starting at states in $L(p)$ that have a position in $L(p)$ greater or equal to that one of p_1.
-it creates a new state p_2 and it creates, for each state r in $L(p)$ that has a position greater or equal to that one of p_1, an edge from r to p_2 with label a
-it creates a path α from p_2 with label the resting part of u. Let p be the end state of this path.
-for each state r' in α, the duplicated of the state r, let a the label of the edge going out from r'. It creates, for each edge $r \xrightarrow{b} s$ with $b \neq a$, the edge $r' \xrightarrow{b} s$.
Preliminaries experiments shows how the automaton is close to the minimal one. It remains an open problem if the constructed automaton is minimal or not.

Time complexity
For each $\mathcal{A}_y^k$, for each state p in $\mathcal{A}_y^k$ we have that $Deg^-(p) \leq (k+1)$. In the indegree control, in the worst case, we have to visit completely the list for the state p with $Deg^-(p) > 1$ and such that $p_1 \neq L(p)[0]$. So for each k, in the worst case, the indegree control takes time $\mathcal{O}(|u| + k)$.

In total the contributions of the visit of the lists $L(p)$ for indegree controls take time $\mathcal{O}(\sum_{k=0,m} k) = \mathcal{O}(|S(y)|)$, so we have that in the worst case the algorithm works in $\mathcal{O}(|S(y)|)$.

6. Conclusion

The algorithm presented in the article builds a small automaton accepting a finite set of words. It has several advantages. It allows an extremely fast compiling of the set of words. With little modification, the method can handle efficiently updates of the automaton, and especially addition of new words. The condition imposed on the list of words is not a restriction because words can always be maintained sorted according to lexicographic order.

One open problem is to find a general upper bound for ratios D (ratio D is the quotient of the number of states of $\mathcal{A}_y$ and of the number of states of its minimal automaton).

Experiments leads us to conjecture that the ratios are bounded by a fixed number, after possibly a small change in the algorithm.

For the suffixes of a word y, we expect that an improved version of the algorithm actually builds the (minimal) suffix automaton of y.

The main open question is whether there exists an on-line construction for the minimal automaton accepting a finite set of words that runs in linear time on each word being inserted in the automaton.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Publishing Company, 1974.
[2] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[3] J. Berstel and D. Perrin. *Theory of Codes*. Academic Press, 1985.

[4] J. Clément, J.-P. Duval, G. Guaiana, D. Perrin, and G. Rindone. Parsing with a finite dictionary. *Theoretical Computer Science*, 340:432–442, 2005.

[5] M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on Strings*. Cambridge University Press, Cambridge, UK,, 2007.

[6] J. Daciuk. Comparison of construction algorithms for minimal, acyclic, deterministic, finite-state automata from sets of strings. In *Proceedings of CIAA 2002*, volume 2608 of *LNCS*, pages 255–261, 2003.

[7] J. Daciuk, S. Mihov, B.W. Watson, and R.E. Watson. Incremental construction of minimal acyclic finite state automata. *Computational linguistics*, 26, number 1:3–16, 2000.

[8] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.

[9] C. Martin-Vide and V. Mitrana. *Grammars and Automata for String Processing: From Mathematics and Computer Science to Biology, and Back*. Taylor and Francis, 2003.

[10] D. Revuz. *Dictionnaires et lexiques: mthodes et algorithmes*. PhD thesis, Institut Blaise Pascal, Paris, France, LITP 91.44, 1991.

[11] D. Revuz. Minimization of acyclic deterministic automata in linear time. *Theoretical Computer Science*, 92, number 1:181–189, 1992.

[12] K.N. Sgarbas, N.D. Fakotakis, and G.K. Kokkinakis. Optimal insertion in deterministic DAWGs. *Theoretical Computer Science*, 301:103–117, 2003.

[13] B. W. Watson. A taxonomy of algorithms for constructing minimal acyclic deterministic finite automata. *South African Computer Journal*, 27:12–17, 2001.

[14] B. W. Watson. A fast and simple algorithm for constructing minimal acyclic deterministic finite automata. *Journal of Universal Computer Science*, 8, number 2:363–367, 2002.

[15] B. W. Watson. A new algorithm for the construction of minimal acyclic DFAs. *Science of Computer Programming*, 48:81–97, 2003.