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

Maxime Crochemore[1] and Laura Giambruno[2]

[1] King's College London, London WC2R 2LS, UK, and Université Paris-Est, France
[2] Dipartimento di Matematica e Applicazioni, Universitàdi Palermo, Palermo, Italy
Maxime.Crochemore@kcl.ac.uk, lgiambr@math.unipa.it

Abstract. 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 the suffixes of a text, showing how this automaton is small. For the suffixes of a text, we propose a modified construction that leads to an even smaller automaton.

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 [6]) and then, using an algorithm for tree minimisation (see [1], [9]), 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 ([11]) 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 Anyway the solution is not completely experimentally tested and remains unpublished.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 [5], [10], [8], [7]). For instance Watson in [15] presented a semi-incremental algorithm for constructing minimal acyclic deterministic automata and Sgarbas et al. in [10] 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 [8] 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 [12] that works in linear time on the size of the acyclic automaton.

Maxime Crochemore, Laura Giambruno: On-line construction of a small automaton for a finite set of words, pp. 15–28.
Proceedings of PSC 2009, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-04403-2 © Czech Technical University in Prague, Czech Republic

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 the list of suffixes of a text. For each set we consider the ratio between the number of states of the constructed automaton and that 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 results in an almost minimal automaton.

In Section 2, after some standard definitions, we define the iterative construction of the automaton for a list of words. 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 bring some examples of the non minimality of this automaton. In Section 4 we deal with sets that are suffixes of a given word. We carry out the experimental results and we show the modified construction. Conclusions are in Section 5.

2 The algorithm for a finite set of words

For definitions on automata we refer to [3] and to [9].

Let A be a finite alphabet. Let x in A^*, then we denote by $|x|$ the length of x, by $x[j]$ for $0 \leq j < |x|$ the letter of index j in x and by $x[j\,..\,k] = x[j] \cdots x[k]$. For any finite set X of words we will denote by $|X|$ the cardinality of 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.
- Indegree: $Deg^-(j)$ is the number of edges ending at j.
- 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 $PF(q_{fin}) = 1$.

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 $X = (\texttt{aaa}, \texttt{ba}, \texttt{aab})$.

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.

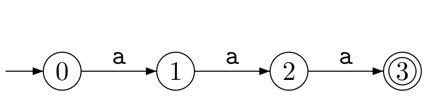

Figure 1. The automata $\mathcal{A}_X^0$ (left) and $\mathcal{A}_X^1$ (right) for $X = (\texttt{aaa}, \texttt{ba}, \texttt{aab})$. Since u, the prefix in common between $\texttt{aaa}$ and $\texttt{ba}$, is the empty word and since s, the suffix in common between $\texttt{aaa}$ and $\texttt{ba}$ is $\texttt{a}$,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

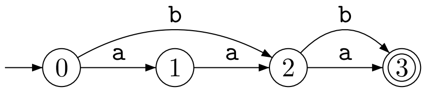

Figure 2. Incorrect construction of $\mathcal{A}_X^2$ for $X = (\texttt{aaa}, \texttt{ba}, \texttt{aab})$: in this case $u = \texttt{aa}$ and $s = \varepsilon$, but, since $Deg^{-}(2) > 1$, adding the edge $(2, \texttt{b}, 3)$ leads to an automaton accepting $\{\texttt{aaa}, \texttt{ba}, \texttt{aab},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

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 $r_0 \xrightarrow{x[\ell]} r$,

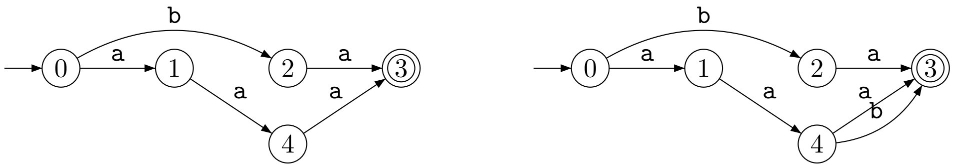

Figure 3. Automata $\mathcal{B}_X^1$ (left) and $\mathcal{A}_X^2$ (right) for $X = ($`aaa,ba,aab`$)$. The automaton $\mathcal{B}_X^1$ is equivalent to $\mathcal{A}_X^1$ and it is obtained from $\mathcal{A}_X^1$ by doing a copy of the path from 0 to 4 with label `aa`. The automaton $\mathcal{A}_X^2$ is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

- we construct a path from r_0 with label $x[\ell]u_1$, let us call p' its ending state,
- 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', $p' \xrightarrow{a} t$.

More formally we define $Q'_{k-1} = Q_{k-1} \cup \{|Q_{k-1}|, \dots, |Q_{k-1}| + |u_1|\}$ and

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

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 $\{x_0, \dots, x_{k-1}\}$ 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 $PF[q] > 1$.

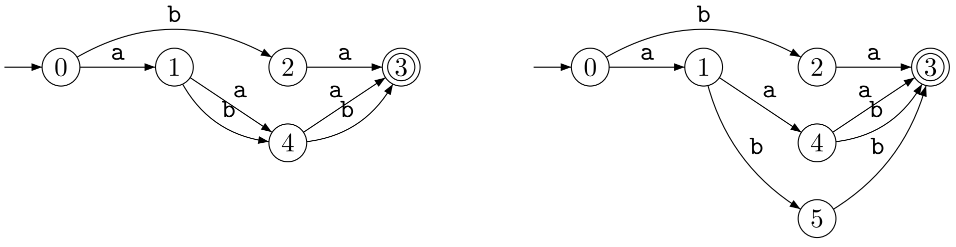

Figure 4. Incorrect construction of $\mathcal{A}_X^3$ (left) for $X = ($`aaa,ba,aab,abb`$)$: adding the edge $(1, \mathtt{b}, 4)$ to $\mathcal{A}_X^2$ leads to an automaton accepting $\{$`aaa,ba,aab,abb,aba`$\}$. Right construction of $\mathcal{A}_X^3$ (right): it is obtained by adding the path from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

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.

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.

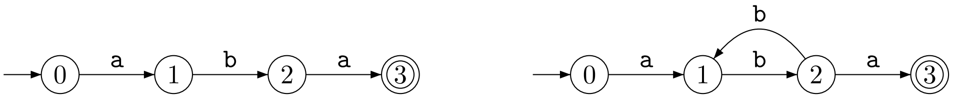

Figure 5. We have the automaton $\mathcal{A}_X^0$ (left) for $X = (\texttt{aba}, \texttt{abbba})$ and the incorrect construction of $\mathcal{A}_X^1$ (right): adding the edge $(2, \texttt{b}, 1)$ would lead to an automaton accepting the infinite language $\{\texttt{aba}, \texttt{a(bb)}^*\texttt{a}\}$.

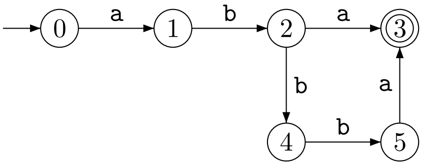

Figure 6. We have the right construction of $\mathcal{A}_X^1$ for $X = (\texttt{aba}, \texttt{abbba})$.

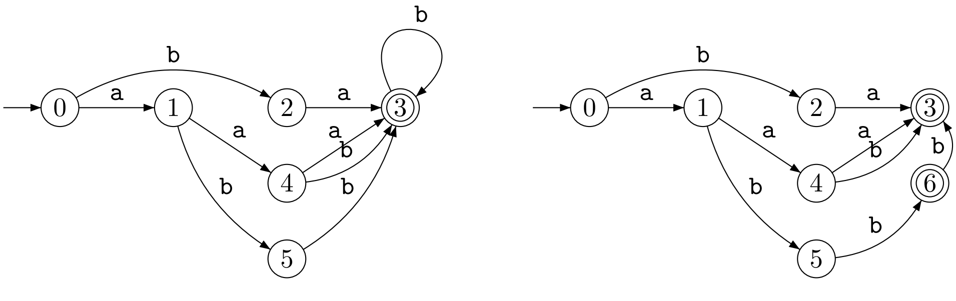

Figure 7. Incorrect construction of $\mathcal{A}_X^4$ (left) and the right construction of $\mathcal{A}_X^4$ (right) for $X = (\texttt{aaa}, \texttt{ba}, \texttt{aab}, \texttt{abb}, \texttt{abbb})$.

If there exists a word in $\{x_0, \ldots, x_{k-1}\}$ that is a prefix of x_k then, if $p \neq q_{fin}$ we add p to the set of final states, that is $T_k = T'_{k-1} \cup \{p\}$.

If $p = q_{fin}$ then, 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.

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 starting from i and ending at the final state q_{fin}.*

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 $\mathcal{A}_X^k$, $0 \leq k \leq m$.

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-$\mathcal{A}_X$($X$)
 1. $(\mathcal{A}, R) \leftarrow$ Construction-$\mathcal{A}_X^0$ $(X[0])$
      $\triangleleft$ denote by $q_{fin}$ the final state of $\mathcal{A}$, define $PF[q_{fin}] = 1$
 2. for $k \leftarrow 1$ to $|X| - 1$ do
 3.       $(\mathcal{A}, R) \leftarrow$ Add-word$(\mathcal{A}, X[k], X[k-1], R)$
 4. Return $\mathcal{A}$
```

Let us consider now the function Construction-$\mathcal{A}_X$. In line 1 we have the function Construction-$\mathcal{A}_0$ that 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.

```
Add-word($\mathcal{A}, x, y, R$)
 1. compute $s$ the suffix in common between $x$ and $y$
 2. $(\mathcal{A}, j, p) \leftarrow$ Indegree-Control $(\mathcal{A}, x)$
 3. if $|x| = j$ then
 4.     $p \leftarrow final(\mathcal{A})$
 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 $|x| - |s| \leq j$ then $s \leftarrow s[j+1\,..\,|s|-1]$
 9.     $q \leftarrow R[|R| - |s|]$
10.     $(\mathcal{A}, q, h) \leftarrow$ PF-Control$(\mathcal{A}, q, s)$
11.     if $PF[q] \neq 1$ then $q_{fin} \leftarrow q$
12.     else
13.         $s \leftarrow s[h\,..\,|s|-1]$
14.         $(\mathcal{A}, q, h) \leftarrow$ Height-Control$(\mathcal{A}, p, q, s)$
15.     if $x[0\,..\,j-1] \in X$ 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.         $p_2 \leftarrow final(\mathcal{A})$
19.     $(\mathcal{A}, R) \leftarrow$ Add path$(\mathcal{A}, x[0\,..\,j-1], x[j\,..\,h-1], q)$
20.Return $(\mathcal{A}, R)$
```

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

```
Indegree-Control($\mathcal{A}, x$)
 1. $p \leftarrow 0$, $j \leftarrow 0$, $InDegControl \leftarrow 0$
 2. while $\delta(p, x[j]) \neq NIL$ and $j \neq |x|$
 3.     $p_1 \leftarrow p$
 4.     $p \leftarrow \delta(p, x[j])$
 5.     if $InDegControl = 0$ then
 6.         if $Deg^-[p] \neq 1$ 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.             $InDegControl \leftarrow 1$
10.     else
11.         create an edge from $p_2$ to a new state $q$ with label $x[j]$
12.         $Deg^-[q] \leftarrow 1$
13.         $p_2 \leftarrow q$
14.     $j \leftarrow j + 1$
15.if $InDegControl = 1$ then
16.     for each edge starting at $p$, with label $a$ and ending state $q$
17.         create an edge from $p_2$ to $q$ with label $a$
18.         $Deg^-[\delta(p_2, a)] \leftarrow Deg^-[\delta(p_2, a)] + 1$
19.     $H[p_2] \leftarrow H[p]$
20.     $p \leftarrow p_2$
21.Return $(\mathcal{A}, j, p)$
```

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 defines the height of p.

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 final states, 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 ($\mathcal{A}, q, s$)
  $h \leftarrow 0$
  while $PF[q] \neq 1$ and $h \neq |s|$
      $q \leftarrow \delta(q, s[h])$
      $h \leftarrow h + 1$
  Return ($\mathcal{A}, q, h$)
```

```
Height-Control ($\mathcal{A}, p, q, s$)
  $h \leftarrow 0$
  while $H[p] \leq H[q]$ and $h \neq |s|$
      $q \leftarrow \delta(q, s[h])$
      $h \leftarrow h + 1$
  Return ($\mathcal{A}, q, h$)
```

In line 15 it controls if $x[0\,..\,j-1]$ is in X. 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 the list 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 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 $X = (\texttt{aaa,ba,aab,bb})$. In order to construct $\mathcal{A}_X^2$ we have to do the indegree control as 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.

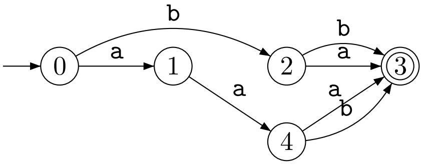

Figure 8. Non minimal automaton $\mathcal{A}_X^3$ for $X = (\texttt{aaa,ba,aab,bb})$.

4 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.

4.1 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 3. *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 $PF(q) = 1$.*

Proposition 4. *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 $H(p) > H(q)$.*

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

Proposition 5. *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 in $\mathcal{A}_y^{k-1}$, let p its final state. If there exists a path from 0 to p with label v in $\mathcal{A}_y^{k-1}$ then, if $|v| < |u|$ then $vz \in \{y_{k+1}, \ldots, y_m\}$, otherwise $vz \in \{y_0, \ldots, y_{k-1}\}$.*

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 6. *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 $|u| > |v|$.*

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 from p_2 with label the resting part of u. Let p be the end state of this path.
- it creates, for each edge, starting at p' an edge starting at p with the same ending state .

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 $\mathcal{O}(|S(y)|)$.

5 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. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman: *Basic local alignment search tool.* J. Mol. Biol., 215 1990, pp. 403–410.
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 2005, pp. 432–442.
5. 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.
6. M. Crochemore, C. Hancart, and T. Lecroq: *Algorithms on Strings*, Cambridge University Press, Cambridge, UK,, 2007.
7. J. Daciuk: *Comparison of construction algorithms for minimal, acyclic, deterministic, finite-state automata from sets of strings*, in Proceedings of CIAA 2002, vol. 2608 of LNCS, 2003, pp. 255–261.
8. J. Daciuk, S. Mihov, B. W. Watson, and R. E. Watson: *Incremental construction of minimal acyclic finite state automata.* Computational linguistics, 26(1) 2000, pp. 3–16.
9. J. E. Hopcroft and J. D. Ullman: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Company, 1979.
10. N. D. F. K. N. Sgarbas and G. K. Kokkinakis: *Optimal insertion in deterministic DAWGs.* Theoretical Computer Science, 301 2003, pp. 103–117.
11. D. Revuz: *Dictionnaires et lexiques: mthodes et algorithmes*, PhD thesis, Institut Blaise Pascal, Paris, France, LITP 91.44, 1991.
12. D. Revuz: *Minimization of acyclic deterministic automata in linear time.* Theoretical Computer Science, 92, number 1 1992, pp. 181–189.
13. B. W. Watson: *A taxonomy of algorithms for constructing minimal acyclic deterministic finite automata.* South African Computer Journal, 27 2001, pp. 12–17.
14. B. W. Watson: *A fast and simple algorithm for constructing minimal acyclic deterministic finite automata.* Journal of Universal Computer Science, 8, number 2 2002, pp. 363–367.
15. B. W. Watson: *A new algorithm for the construction of minimal acyclic DFAs.* Science of Computer Programming, 48 2003, pp. 81–97.

6 Appendix

Proof of Lemma 2

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. If i is a state of $\mathcal{A}_X^k$ with $Deg^-(i) > 1$ 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 to 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 Lemma 2, 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 $\mathcal{A}_k$ 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 $Deg^-(p) = 1$ 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 $\mathcal{A}_k$ 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.

Proof of Proposition 3

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 $\{y_0, \ldots, y_{k-1}\}$.

Proof of Proposition 4

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. 3, 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 $\{y_0, \ldots, y_{k-1}\}$.

Proof of Proposition 5

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 $\{y_0, \ldots, y_{k-1}\}$.

Proof of Proposition 6

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 $|v| < |u|$.