Combinatorial Characterization of the Language Recognized by Factor and Suffix Oracles

Alban Mancheron, Christophe Moan

International Journal of Foundations of Computer Science

COMBINATORIAL CHARACTERIZATION OF THE LANGUAGE RECOGNIZED BY FACTOR AND SUFFIX ORACLES

ALBAN MANCHERON

and

CHRISTOPHE MOAN

Laboratoire d'Informatique de Nantes-Atlantique,
Université de Nantes,
B.P. 92 208, 44 322 Nantes Cedex 3, France

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

Sequence Analysis requires to elaborate data structures, which allow both an efficient storage and use. A new one was introduced in 1999 by Cyril Allauzen, Maxime Crochemore and Mathieu Raffinot. This structure is linear on the size of the represented word both in time and space. It has the smallest number of states and it accepts at least all substrings of the represented word. This structure is called *Factor Oracle*. Authors developed another structure on the basis of Factor Oracle, which has the same properties except it accepts at least all suffixes instead of all factors of the represented word. This structure is then called *Suffix Oracle*. The characterization of the language recognized by the Factor/Suffix Oracle of a given word is an open problem, for which we provide a solution. Using this result, we show that these structures may accept an exponential number of words, which are not factors/suffixes of the given word.

Keywords: Factor Oracle, Suffix Oracle, automata, language, characterization.

1. Introduction

Several structures have been developed in text indexation: we can cite Tries [1], Suffix Automata [1, 2], Suffix Trees [1, 3]... Their objective is to represent a text or a word s (*i.e.* a succession of symbols taken in an arbitrary alphabet denoted by Σ), in order to "quickly" determine whether this word contains some specific sub-word. This sub-word is then called a factor of s.

Allauzen *& al.* [4, 5, 6] described a method allowing to build an *acyclic* automaton, which accepts at **least** all factors of s, which have as few states as possible ($|s|+1$) and which is *linear* in the number of transitions ($2|s|-1$). When each state is final in this automaton, the structure is called a *Factor Oracle*. By keeping only "particular" states as final, this automaton becomes a *Suffix Oracle*.

Several advantages have been described for this structure. The construction algorithm is easy to understand and implement; such advantage doesn't exist in the most efficient algorithm to build Suffix Trees [3]. Oracles are homogeneous automata, *i.e.* all transitions ingoing to a same state are labeled with a same symbol. Thus, it is not necessary to label edges anymore. Therefore this structure requires less memory than Suffix Trees or Tries. LEFEBVRE & *al.* [7, 8, 9] used it for repeated motifs discovery over large genomic data and obtained in a few seconds similar results to the ones obtained by using thousands of BLASTn requests. Authors also used the Factor Oracle for text compression [10].

However, at least two open questions are linked to these Oracles: the first one is about the characterization of the language recognized by Oracles; and the second question concerns the existence of a linear algorithm in time and space to build an automaton, which accepts all factors/suffixes of a word s and which is minimal in number of transitions.

When using these Oracles, the main difficulty is to distinguish true and false positives. Therefore we will provide in the next section several definitions related to the construction of Oracles. We will characterize the language recognized by this structure in section 3. Finally, some results using Oracles will also be described.

2. Definitions

In the following sections, we call $Fact(s)$ (resp. $Suff(s)$ and $Pref(s)$) the set of factors (resp. suffixes and prefixes) of $s \in \Sigma^+$. We call $Pref_s(i)$ the prefix of s, which has length $i \geq 0$. Given $x \in Fact(s)$, $Nb_s(x)$ is the number of occurrences of x in s and x is repeated if and only if $Nb_s(x) \geq 2$.

Definition 1 *Given a word $s \in \Sigma^+$ and x a factor of s, we define the function Pos as the position of the first occurrence of x in $s = uxv$ $(u, v \in \Sigma^*)$: $Pos_s(x) = |u|+1$. We also define the function poccur such that $poccur_s(x) = |ux| = Pos_s(x)+|x|-1$.*

2.1. Oracles

The Oracle construction is defined by the algorithm of ALLAUZEN & *al.* [4] (see algorithm 1). Authors gave another algorithm to build the same automaton in linear time on the size of s. However, since we are only interested in properties of the Oracle, we do not report it in this paper.

Definition 2 *[4] Given a word $s \in \Sigma^*$, we define the Factor Oracle of s as the automaton obtained by the algorithm 1, where all states are final. It is denoted by $FO(s)$. We define the Suffix Oracle of s as the automaton obtained by the same algorithm, where a state e_i $(0 \leq i \leq |s|)$ is final if and only if there exists a path labeled by a suffix of s from the initial state to the state e_i. It is denoted by $SO(s)$.*

We use the term Oracle to equally designate the Factor or the Suffix Oracle of a word s and we denote it by $O(s)$. We define a relation of order between states in these Oracles. Indeed, if we have two states e_i and e_j such that $i \leq j$, then $e_i \leq e_j$.

[a] As mentioned in [11], the term $-|u|$ (line 17) is unfortunately missing in the original algorithm.

Algorithm 1: Construction of the Factor Oracle of a word[a]

```
Input: Σ % Alphabet (supposed minimal) %
       s ∈ Σ* % The word to process %
Output: Oracle % Factor Oracle of s %

Begin
  Create the initial state labeled by e_0

  For i from 1 to |s| Do
    Create a state labeled by e_i
    Build a transition from the state e_{i-1} to the state e_i labeled by s[i]
  End For

  For i from 0 to |s| − 1 Do
    Let u be a word of minimal length recognized at state e_i
    For All α ∈ Σ \ {s[i + 1]} Do
      If uα ∈ Fact(s[i − |u| + 1..|s|]) Then
        j ← poccur_{s[i−|u|+1..|s|]}(uα) − |u|
        Build a transition from the state e_i to e_{i+j} labeled by α
      End If
    End For All
  End For
End
```

Definition 3 *Given a word $s \in \Sigma^*$ and a word x accepted at the state e_i $(0 \leq i \leq |s|)$ in the Oracle of s, we define the function State as $State(x) = e_i$.*

Lemma 1 *[4] Given a word $s \in \Sigma^*$, a unique word with minimal length is accepted at each state e_i $(0 \leq i \leq |s|)$ in the Oracle of s. It is denoted by $min(e_i)$.*

Lemma 2 *[4] Given a word $s \in \Sigma^*$, its Oracle and an integer i $(0 \leq i \leq |s|)$, then $min(e_i) \in Fact(s)$ and $i = poccur_s(min(e_i))$.*

Notation 1 *Given a word $s \in \Sigma^*$, let $\#_{in}(e_i)$ and $\#_{out}(e_i)$ denote the number of ingoing/outgoing transitions to/from state e_i $(0 \leq i \leq |s|)$ in the Oracle of s.*

2.2. Canonical Factors & Contraction Operation

In this section, we will define particular factors from a given word and an operation needed to characterize the language. Then we will define the sets of words obtained with this operation.

Definition 4 *Given a word $s \in \Sigma^*$ and its Oracle, we define the set of Canonical Factors of s as $\mathcal{F}_s = \{min(e_i) \,|\, 1 \leq i \leq |s| \wedge (\#_{out}(e_i) > 1 \vee \#_{in}(e_i) > 1)\}$. Given a suffix t of s and a Canonical Factor f of s, f is a conserved Canonical Factor of s in t if the first occurrence of f in s appears in t. The set of the conserved Canonical Factors of s in t is denoted by $\mathcal{F}_{s,t}$ (thus $\mathcal{F}_{s,t} \subseteq \mathcal{F}_s$).*

Definition 5 *Given a word $s \in \Sigma^*$ and a repeated Canonical Factor f of s such that:*

$$\begin{cases} s & = & ufv & (u, v \in \Sigma^*) \\ fv & = & wfx & (w \in \Sigma^+, x \in \Sigma^*) \\ Pos_s(f) & = & |u| + 1 & \end{cases}$$

then the pair $(|u| + 1, |uw| + 1)$ is a contraction of s by f.

Notation 2 *Given a word $s \in \Sigma^*$ and a Canonical Factor $f \in \mathcal{F}_s$, $\mathcal{C}_s^f$ is the set of contractions of s by f and $\mathcal{C}_s^*$ ($\equiv \bigcup_{f \in \mathcal{F}_s} \mathcal{C}_s^f$) is the set of all the contractions we can apply to s. Given a suffix t of $s = t't$ $(t, t' \in \Sigma^*)$, then $\mathcal{C}_{s,t}^*$ is the subset of $\mathcal{C}_s^*$ such that $\mathcal{C}_{s,t}^* = \{(p,q) \mid (p,q) \in \mathcal{C}_s^* \ \wedge \ p > |t'|\}$.*

In the following, we will use sets of contractions to produce new words from a given one. Then we will use these words to characterize the language recognized by Oracles.

Definition 6 *A set $\mathcal{C}$ of contractions is coherent if and only if it does not contain two contractions (i_1, j_1), (i_2, j_2) such that: $i_1 < i_2 < j_1 < j_2$. Furthermore, $\mathcal{C}$ is minimal if and only if it does not contain two contractions (i_1, j_1) and (i_2, j_2) such that $i_1 \leq i_2 < j_2 \leq j_1$ or such that $i_1 < j_1 = i_2 < j_2$.*

Definition 7 *Given a word $s \in \Sigma^*$ and a coherent and minimal set of contractions $\mathcal{C} = \{(p_1, q_1), \ldots, (p_k, q_k)\}$ (associated to the set of canonical factors $\{f_1, \ldots, f_k\}$), then we define the function $Word$ as following:*

$$Word(s, \mathcal{C}) = s[1..p_1 - 1]\, s[q_1..p_2 - 1] \ldots s[q_{k-1}..p_k - 1]\, s[q_k..|s|].$$

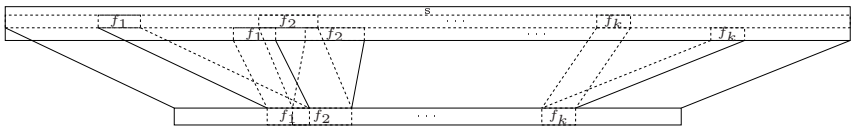

Fig. 1. Words obtained using the contraction operation (see Definition 5).

We are only interested by words obtained by the contraction operation. So we will only consider coherent and minimal sets of contractions without loss of generality. Note that the word remains the same whatever the order of contractions (see figure 1).

Definition 8 *We define $\mathcal{E}(s) = \bigcup_{\mathcal{C} \subseteq \mathcal{C}_s^*} Word(s, \mathcal{C})$, as the closure of s.*

Example: Consider the word *gaccattctc* (see figure 2). Its set of Canonical Factors is $\mathcal{F}_{gaccattctc} = \{a, c, ca, t, tc, ct\}$ and then $\mathcal{C}_{gaccattctc}^* = \{(2,5), (3,4), (3,8), (3,10), (6,7), (6,9), (7,9)\}$. Now, consider the set $\mathcal{C} = \{(2,5), (7,9)\}$ ($\mathcal{C} \subseteq \mathcal{C}_{gaccattctc}^*$), then $Word(gaccattctc, \mathcal{C}) = g\not{a}\not{c}\not{c}\boldsymbol{att}\not{c}\boldsymbol{tc} = gattc$. The closure of *gaccattctc* is:

$$\mathcal{E}(gaccattctc) = \left\{ \begin{array}{l} gac, gacatc, gacatctc, gacattc, gacattctc, gaccatc, gaccatctc, \\ gaccattc, gaccattctc, gactc, gatc, gatctc, \boldsymbol{gattc}, gattctc \end{array} \right\}.$$

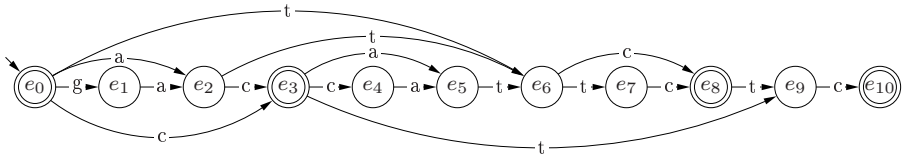

Fig. 2. Suffix Oracle of *gaccattctc*. In the Factor Oracle, all states are final.

3. Characterization of the language recognized by Oracles

Given a word $s \in \Sigma^*$, we saw how to build the corresponding Factor (resp. Suffix) Oracle. This Oracle allows to recognize the factors (resp. suffixes) of s, but it also accepts additional words. For example the word *atc* is accepted by the Factor (resp. Suffix) Oracle of *gaccattctc* (see figure 2), whereas it is neither a factor nor a suffix of this sequence. We will see that the Suffix Oracle of s exactly recognizes all suffixes of words from $\mathcal{E}(s)$ and we will use this result to prove that the Factor Oracle of s exactly recognizes all factors of words from $\mathcal{E}(s)$.

We use following Lemmas to prove the result concerning Suffix Oracles. These Lemmas have been proved in [4].

Lemma 3 *[4] Given a word $s \in \Sigma^*$ and an integer i $(0 \leq i \leq |s|)$, then $min(e_i)$ is suffix of all words recognized at state e_i in the Oracle of s.*

Lemma 4 *[4] Given a word $s \in \Sigma^*$ and a factor w of s, then w is recognized at state e_i $(1 \leq i \leq poccur_s(w))$ in the Oracle of s.*

Lemma 5 *[4] Given a word $s \in \Sigma^*$ and an integer i $(0 \leq i \leq |s|)$, then every path ending by $min(e_i)$ in the Oracle of s leads to a state e_j such that $j \geq i$.*

Lemma 6 *[4] Given a word $s \in \Sigma^*$ and $w \in \Sigma^*$ a word accepted at state e_i $(0 \leq i \leq |s|)$ in the Oracle of s, then every suffix of w is also recognized by the Oracle at state e_j such that $j \leq i$.*

The proof of Lemma 6 is only given in [4] for Factor Oracles. We need this result to be true for Suffix Oracles.

Proof. The original Lemma gives us that if x is a suffix of w, then $State(x) \leq State(w)$. We need to prove that if $State(w)$ is final, then $State(x)$ is final. Therefore, we have to consider two cases. When $|x| \geq |min(e_i)|$, we have $min(e_i) \in Suff(x)$ and thus, according to Lemma 5, $State(x) \geq State(min(e_i))$. Since $State(min(e_i)) = e_i = State(w)$, we conclude that $State(x) = State(w)$. When we have $|x| < |min(e_i)|$, since the state e_i is final, then there exists a suffix t of s such that $State(t) = e_i$. According to Lemma 3, we conclude that $min(e_i) \in Suff(t) \subseteq Suff(s)$. Since x and $min(e_i)$ are suffixes of w, then $|x| < |min(e_i)| \Rightarrow x \in Suff(min(e_i))$. So x is also suffix of s and according to Definition of the Suffix Oracle $State(x)$ is final. □

We use these previous results to show that the two following Lemmas.

Lemma 7 *Given a word $s \in \Sigma^*$, a Canonical Factor f of $s = ufv$ $(u, v \in \Sigma^*)$ such that f is not repeated in uf and a set $\mathcal{C}$ of contractions $(\mathcal{C} \in \mathcal{C}_s^*)$ such that $Word(uf, \mathcal{C}) = wf$, then the Oracle of s accepts wf and f at the same state.*

Proof. We denote by $\mathcal{C}_i \subseteq \mathcal{C}_s^*$ a set of contractions, which has cardinality i. In the same way, we denote by $w_i f$ the word obtained when we apply contractions $\mathcal{C}_i$ to uf (warning: $w_i f = Word(uf, \mathcal{C}_i) \not\Rightarrow w_i = Word(u, \mathcal{C}_i)$). By induction on the size of $\mathcal{C}_i$, we show that $State(Word(uf, \mathcal{C}_i)) = State(f)$ for all $\mathcal{C}_i \in \mathcal{C}_s^*$.
Let $e_x = State(f)$ ($f = min(e_x)$ by Definition of f) and $e_{x'_i} = State(Word(uf, \mathcal{C}_i))$. If we consider $\mathcal{C}_0$, then $Word(uf, \mathcal{C}_0) = uf$. According to Lemma 5, $x'_0 \geq x$. Furthermore, according to Lemma 4 applied to uf, we have $x'_0 \leq poccur_s(uf)$.

However by Definition of f, $poccur_s(f) = |uf| = poccur_s(uf)$. Therefore we have $x'_0 \leq x$ and $x'_0 = x$.
If this lemma is true for a set $\mathcal{C}_i \subset \mathcal{C}_s^*$ of contractions, then it is true for a set $\mathcal{C}_{i+1} = \mathcal{C}_i \cup \{(p,q)\}$. We assume without loss of generality (see figure 1) that (p,q) is the last contraction in $\mathcal{C}_{i+1}$ (by ascending order over positions). Let b be the Canonical Factor used by this contraction. We can write $uf = s[1..p-1]\,s[p..q-1]\,s[q..|uf|]$. Since (p,q) is chosen as the last contraction, all contractions in $\mathcal{C}_i$ are applicable to $s[1..p-1]$. So we define $a, c \in \Sigma^*$ such that $w_i f = a\,s[p..|uf|] = abc$ and $d \in \Sigma^*$ such that $w_{i+1} f = a\,s[q..|uf|] = abd$. Also $ab = Word(s[1..p-1]\,b, \mathcal{C}_i)$: the opposite would mean that contraction (p,q) can't be operated from b and according to the induction hypothesis, $State(ab) = State(b)$. From this, we conclude that $State(abc) = State(bc)$ and $State(abd) = State(bd)$. We know that bd $(= s[q..|uf|])$ is a suffix of bc $(= s[p..|uf|])$ and according to Lemma 6:

$$\begin{array}{lrcl} & State(bd) & \leq & State(bc) \\ \Leftrightarrow & State(abd) & \leq & State(abc) \\ \Leftrightarrow & State(w_{i+1}f) & \leq & State(w_i f) \\ \Leftrightarrow & State(w_{i+1}f) & \leq & State(f) \end{array}$$

However, according to Lemma 5, we have $State(w_{i+1}f) \geq State(f)$ and therefore $State(w_{i+1}f) = State(f)$. This lemma is then true for all $\mathcal{C}_i \subseteq \mathcal{C}_s^*$. □

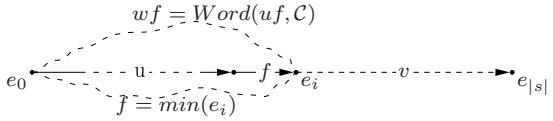

Fig. 3. Illustration of Lemma 7.

Now, we show how to obtain a contraction, starting with transition of type $e_i \rightarrow e_j$ with $j > i+1$ (that we call an external transition in the subsequent).

Lemma 8 *Given a word $s \in \Sigma^*$ and an integer i $(0 \leq i < |s|)$ such that $\#_{out}(e_i) > 1$, let p be a external transition from state e_i to state e_{i+j} $(j > 1)$ labeled by α and $u = min(e_i)$. Then an occurrence of $u\,\alpha$ exists at position $(i + j - |u|)$ of s (see figure 5 – page 8). Moreover, the contraction $(i - |u| + 1, i - |u| + j)$ of s by u exists too.*

Proof. According to the construction algorithm (see algorithm 1), the transition p is added from e_i to e_{i+j} because a position j in $s[i - |u| + 1..|s|]$ is such that: $j = poccur_{s[i-|u|+1..|s|]}(u\alpha) - |u|$. We also have $u\alpha \in Fact(s)$ because $u\alpha \in Fact(s[i - |u|+1..|s|])$. Cleophas & al. [11] proved that since $u = min(e_i)$ and $u\alpha \in Fact(s)$, then $i - |u| + poccur_{s[i-|u|+1..|s|]}(u\alpha) = poccur_s(u\alpha)$. We have $i + j = poccur_s(u\alpha)$ and finally $s[i + j - |u|, i + j] = u\alpha$. □

We use algorithm Contractor (see algorithm 2) to give a characterization of the language accepted by the Oracle of a word s. Given a word $s \in \Sigma^*$ and its Suffix Oracle $SO(s)$, initial inputs of Contractor are a word w accepted by $SO(s)$ and t, the maximal suffix of s beginning with $w[1]$. This algorithm outputs a set $\mathcal{C} \in \mathcal{C}_s^*$ of contractions such that $w = Word(t, \mathcal{C})$.

Algorithm 2: Contractions needed to transform $s = t't$ $(t, t' \in \Sigma^*)$ into $t'w$

```
1  Initialization: S_0 = t, S_0^w = w, C_0 = ∅, sdec = |s| − |t| % t is a suffix of s %
2
3  Input:  S_i ∈ Σ* % A suffix of s that can still be "contracted" %
4          S_i^w ∈ Σ* % The word to process %
5          C_i % Set of contractions %
6  Output: a set of contractions
7
8  Begin
9    p_i ← longest common prefix between S_i and S_i^w                    (Claim 1, item 1)
10   e_{q_i} ← State(p_i)                                                 (Claim 1, item 2)
11   f_i ← min(e_{q_i})
12   If (|p_i| < |S_i^w|) Then
13     e_{r_i} ← Transition(e_{q_i}, S_i^w[|p_i| + 1])                    (Claim 1, item 4)
14     C_{i+1} ← C_i ∪ {c_i}, c_i = (q_i − |f_i| + 1, r_i − |f_i|)        (Claim 2, item 2)
15     S_{i+1}^w ← S_i^w[|p_i| − |f_i| + 1..|S_i^w|]                      (Claim 1, item 3)
16     S_{i+1} ← t[r_i − |f_i| − sdec..|t|]                               (Claim 1, item 3)
17     Return Contractor(S_{i+1}, S_{i+1}^w, C_{i+1})
18   Else
19     If (|S_i| > |S_i^w|) Then
20       C_{i+1} ← C_i ∪ {c_i}, c_i = (q_i − |f_i| + 1, |s| − |f_i| + 1)  (Claim 3)
21     Else
22       C_{i+1} ← C_i                                                    (Claim 3)
23     End If
24     Return C_{i+1}
25   End If
26 End
```

By Definition 7, given a word $s \in \Sigma^*$ such that $s = t't$ $(t, t' \in \Sigma^*)$ and a set $C \in \mathcal{C}^*_{s,t}$ of contractions, $t'w = Word(s, C)$ is then a concatenation of substrings of s. These substrings can be assimilated as prefixes of suffixes of s. A contraction is then a "jump" from one substring to the next one. The main idea of this algorithm is to read from left to right the sequences t and w, in order to compute the longest common prefixes between given suffixes of t and w. After each stage, the algorithm computes the length of the "jump" to go to the next suffix of t to consider.

Inputs are words S_i, S_i^w $(i \geq 0)$ and a set $\mathcal{C}_i$ of contractions. We initialize $S_0 = t$, $S_0^w = w$, $\mathcal{C}_0 = \emptyset$ and p_i (line 9) as the longest prefix of S_i and S_i^w. So:

$$S_i = p_i S'_i \qquad \text{and} \qquad S_i^w = p_i S_i'^w. \tag{1}$$

Let $e_{q_i} = State(p_i)$ and $f_i = min(e_{q_i})$ ((lines 10 and 11), Lemma 3 provides:

$$p_i = p'_i f_i \quad (p'_i \in \Sigma^*). \tag{2}$$

The variable e_{r_i} (line 13) is the state reached by the transition from e_{q_i} and labeled by $\alpha = S_i^w[|p_i| + 1] = S_i'^w[1]$. The set $\mathcal{C}_{i+1}$ has cardinality $i + 1$. The variable $sdec = |s| - |t|$ is necessary to compute S_{i+1} (line 15), because each state e_i is linked to the i^{th} character of s, not to the character $(i - |s| + |t|)$ of t.

Claim 1 *Following assertions are true for all $i \geq 0$ (see figure 4):*

1. $f_i \alpha \in Pref(p_{i+1})$.
2. $S_i = t[q_i - |p_i| + 1 - sdec..|t|]$.
3. *S_{i+1} and S_{i+1}^w are respectively suffixes of S_i and S_i^w; S_i and S_i^w $(i \geq 0)$ are respectively suffixes of t and w.*
4. *Transition from e_{q_i} to e_{r_i} labeled by α always exists.*

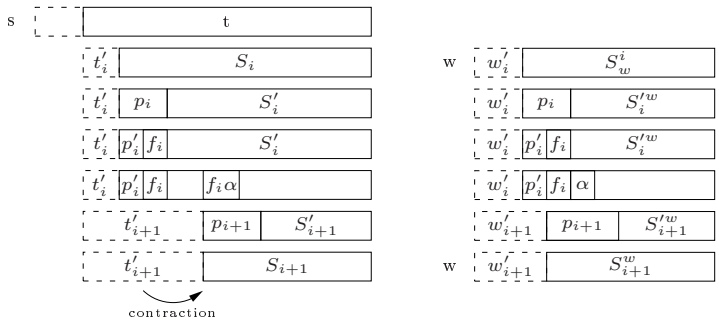

Fig. 4. Visualization of Contractor on S_i and S^w_i.

Proof.

1. Since $f_i = min(e_{q_i})$ and according to Lemma 8, we have $s[r_i - |f_i|..r_i] = t[r_i - |f_i| - sdec..r_i - sdec] = f_i\alpha$. So S_{i+1} and S^w_{i+1} begin with $f_i\alpha$ (line 15).

2. For $i = 0$ (initialization case), $S_0 = t$ and t is the longest suffix of s beginning by $w[1]$. If $e_x = State(S_0[1])$ $(x > 0)$, then $t[x - sdec..|t|] = S_0$ and $State(p_0) = x + |p_0| - 1 = e_{q_i}$. Thus we conclude that $S_0 = s[q_0 - |p_0| + 1 - sdec..|s|]$. At iteration i, we have $S_{i+1} = t[r_i - |f_i| - sdec..|t|]$ (line 16). Since S^w_{i+1} begins by $f_i\alpha$ (item 1), we conclude that $q_{i+1} = r_i + |p_{i+1}| - |f_i| - 1$ and finally we have $S_{i+1} = t[r_i - |f_i| - sdec..|t|] = t[q_{i+1} - |p_{i+1}| + 1 - sdec..|t|]$.

3. This assertion is true for S^w_i because S^w_{i+1} is suffix of S^w_i by construction (line 15) and $S^w_0 = w$. Concerning S_i, we have $S_0 = t$ thus the property is true for $i = 0$. Suppose that S_i is suffix of t, from item 2, we have $S_i = t[q_i - |p_i| + 1 - sdec..|t|]$. We also have $S_{i+1} = t[r_i - |f_i| - sdec..|t|]$ (line 16). So we conclude using Eq. (2) that $q_i - |p_i| = q_i - |p'_i| - |f_i|$. Since $|p'_i| \geq 0$, we have $q_i - |f_i| \geq q_i - |p_i|$ and $r_i > q_i$. Finally, we conclude that $r_i - |f_i| > q_i - |p_i|$ and that S_{i+1} is a suffix of S_i.

4. According to item 3, S^w_i is suffix of w. Then S^w_i is accepted by $O(s)$ (Lemma 6). Using Eq. (1) with $S'^w_i[1] = \alpha$ the transition must exist and implies $\#_{out}(e_{q_i}) \geq 2$. So $f_i = min(e_{q_i}) \in \mathcal{F}_s$ by Definition of Canonical Factors. □

From Eq. (1) and Claim 1 (item 4):

$$\left\{ \begin{array}{rcccll} t & = & t'_i S_i & & & (t'_i \in \Sigma^*) \\ w & = & w'_i S^w_i & = & w'_i p'_i S^w_{i+1} & (w'_i \in \Sigma^*). \end{array} \right. \tag{3}$$

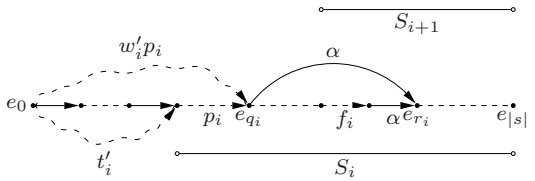

Fig. 5. Illustration of a step in the algorithm Contractor ($\alpha = S^w_i[|p_i| + 1]$).

Claim 2 *For all $i \geq 0$ (see figure 5):*

1. *$State(w'_i p_i) = State(t'_i p_i) = State(p_i) = e_{q_i}$.*
2. *c_i is a contraction of $t'_i S_i$ (resp. $w'_i S_i$) provided by f_i. Result of this contraction is $t'_i p'_i S_{i+1}$ (resp. $w'_i p'_i S_{i+1} = w'_{i+1} S_{i+1}$).*

Proof.

1. For $i = 0$, $t'_i = w'_i = \epsilon$. Let us suppose that the property is true for i and therefore for $i+1$. From Claim 1 (item 2), we conclude that S_i labeled a path using only "main" transitions (*i.e.* transitions of type $e_j \rightarrow e_{j+1}$) from $e_{q_i-|p_i|}$ to $e_{|s|}$ in $O(s)$. According to Claim 1 (item 3) we conclude that:

$$S_i = uS_{i+1} \quad (u \in \Sigma^*). \tag{4}$$

So, the state e_x ($x > q_i - |p_i|$) is such that S_{i+1} labeled a path using only "main" transitions from e_x to $e_{|s|}$ in $O(s)$ and more precisely $x = r_i - |f_i| - 1$. We have $t'_{i+1} = t'_i u$ (Eq. (3) and (4)) and $State(t'_i u) = e_x$. Since $f_i = min(e_{q_i})$ and there is a transition from e_{q_i} to e_{r_i} labeled by α (Claim 1, item 4), $State(t'_{i+1} f_i \alpha) = State(t'_i u f_i \alpha) = State(f_i \alpha) = e_{r_i}$ and furthermore, $p_{i+1} = f_i \alpha v$ ($v \in \Sigma^*$). So we have $State(t'_{i+1} f_i \alpha v) = State(t'_{i+1} p_{i+1}) = State(p_{i+1})$.

2. From Eq. (1), (2) and (3), we conclude that:

$$t = t'_i S_i = t'_i p'_i f_i S'_i. \tag{5}$$

Since $S_{i+1} \in Suff(S_i)$, we have $S_i = uS_{i+1}$ ($u \in \Sigma^+$) and from Eq. (5), $t'_i p'_i f_i S'_i = t'_i u S_{i+1}$. According to Claim 1 (item 1), we have $t'_i p'_i f_i S'_i = t'_i u f_i \alpha u'$ ($u' \in \Sigma^*$). Since we have $State(t'_i p'_i f_i) = State(f_i)$ and $|u| > |p'_i|$ (because $S'_i[1] \neq \alpha$), we can contract $t'_i S_i$ by f_i and $t'_i p'_i f_i \alpha u' = t'_i p'_i S_{i+1}$. We can conclude that $w'_i S_i = w'_i p_i S'_i$ is contracted by f_i in $w'_i p'_i S_{i+1}$ because $State(w'_i p_i) = State(t'_i p_i)$. According to Eq. (3), we conclude that $w'_{i+1} = w'_i p'_i$ and $w'_i p'_i S_{i+1} = w'_{i+1} S_{i+1}$. □

Claim 2 shows that a contraction c_i of $t'_i S_i$ by f_i is also a contraction for t and for $w'_i S_i$. Consider the i^{th} iteration, we have $|S^w_i| > |S^w_{i+1}|$ or $|S^w_i| = |S^w_{i+1}|$ and $|p_{i+1}| > |p_i|$ (if $f_i = p_i$). Since $p_i > 0$, we ensure that recursion stops at iteration $j > i$ with $p_j = S^w_j$ ($j > i$).

Claim 3 *Given an integer $i \geq 0$ such that $p_i = S^w_i$, then t needs a last contraction if and only if $|S^w_i| \neq |S_i|$.*

Proof. The word obtained with the set $\mathcal{C}_i$ of contractions is $w'_i p_i S'_i$ (Claim 2, item 2). If $S^w_i = S_i$, then we have $S'_i = \epsilon$ and we conclude that $\mathcal{C}_i$ is complete (line 20). If $S^w_i \neq S_i$, according to Claim 2 (item 1), we have $State(w'_i p_i) = e_{q_i}$. By Definition of the final state in a Suffix Oracle, $min(e_{q_i}) \in Suff(w)$ (Lemma 3) and $min(e_{q_i}) \in Suff(t)$. A last contraction is needed to complete the set of contractions (line 22). □

Considering the i^{th} call of Contractor, inputs are $S_i = p_i S'_i$, $S^w_i = p_i S'^w_i$ and $\mathcal{C}_i$. This set is the set of contractions necessary to transform $t'_i \in Pref(t)$ into $w'_i \in Pref(w)$. The variable p_i refers to the longest common prefix of S_i and S^w_i (p_i is both a factor of t and w, Claim 1). Two cases may occur for this call (line 12. When $|p_i| = |S^w_i|$, the recursion ends. Otherwise, $|p_i| \neq |S^w_i|$ and at least another contraction is necessary until $|p_j| = |S^w_j|$ ($j > i$). From Claim 2, the i^{th} contraction allows to continue with the suffix S_{i+1}. At the end of the process (*i.e.* the end of w), we return the set $\mathcal{C} = \mathcal{C}_j$ and Lemma 9 ensures that $w = Word(t, \mathcal{C})$.

Lemma 9 *Given a word $s \in \Sigma^*$, its Suffix Oracle, a word $w \in \Sigma^*$ accepted by $SO(s)$ and the longest suffix t of s such that $w[1] = t[1]$, then Contractor$(t, w, \emptyset)$ outputs a set $\mathcal{C}$, which is such that $w = Word(t, C)$.*

Proof. Let $i \geq 0$ such that $S^w_{i+1} = p_{i+1}$. According to Claim 2, we conclude that $\mathcal{C}_{i+1}$ is a coherent set of contractions of t. Since p_{i+1} is prefix of S_{i+1} we have:

$$Word(t, \mathcal{C}_{i+1}) = w'_{i+1} S_{i+1} = w'_{i+1} S^w_{i+1} u = w'_{i+1} p_{i+1} u \quad (u \in \Sigma^*)$$

If $u = \epsilon$, we have $Word(t, \mathcal{C}_{i+1}) = w$ (Eq. (3)). Else (Claim 3) a ultimate contraction c_{i+1} by f_{i+1} transforms $w'_{i+1} S^w_{i+1} u$ into $w'_{i+1} S^w_{i+1} = w = Word(t, \mathcal{C}_{i+1} \cup \{c_{i+1}\})$. Finally Contractor provides a set $\mathcal{C}$ such that $w = Word(t, \mathcal{C})$. □

We can notice that:

1. $\mathcal{C}$ is not always minimal.
2. $\mathcal{C}$ is coherent. Let (a, b) and (c, d) be two contractions successively added to $\mathcal{C}$. We have $a < b$ and $c < d$ because $r_i > q_i$ and $|s| > q_i$ (lines 14 and 20). If $e_{q_{i+1}} = State(p_{i+1}) = e_{r_i}$, then $b = c$, else $p_{i+1} = f_i \alpha v$ ($\alpha = S^w_{[}|p_i|+1], v \neq \epsilon$) and $e_{q_{i+1}} > e_{r_i}$, therefore $b < c$.

Following theorems are the main purpose of this paper:

Theorem 1 *Exactly all suffixes of words from $\mathcal{E}(s)$ are recognized by the Suffix Oracle of s .*

Proof.
'⇒': *Each suffix of words from $\mathcal{E}(s)$ is recognized by the Suffix Oracle of s.*
According to Lemma 6, if w is accepted by $SO(s)$, each suffix of w is also accepted by $SO(s)$. So we only need to prove that each word from $\mathcal{E}(s)$ is accepted by $SO(s)$. Let $\mathcal{C} \in \mathcal{C}^*_s$ be a set of contractions applicable to s and $w = Word(s, \mathcal{C})$. The set $\mathcal{C}_i$ is the set of the first i contractions of $\mathcal{C}$ (chosen without loss of generality by ascending order over positions – see figure 1), (x_j, y_j) is the j^{th} contraction, which use the Canonical Factor f_j ($1 \leq j \leq i$) and $w_j = Word(s, \mathcal{C}_j)$. The property (P) to prove is that if w_i ($0 \leq i < |\mathcal{C}|$) is accepted by $SO(s)$, then w_{i+1} is accepted too. We have:

$$\begin{cases} w_i & = & s[1..x_1 - 1]\, s[y_1..x_2 - 1] \ldots s[y_i..|s|] \\ s[y_i..y_i + |f_i| - 1] & = & f_i \end{cases}$$

By Definition of the Canonical Factors, f_{i+1} does not occur in s before position x_{i+1} ($x_{i+1} > y_i$). We have $w_i = v' f_{i+1} u$ and $w_{i+1} = v' f_{i+1} u'$ with $v' = s[1..x_1 - 1] s[y_1..x_2 - 1] \ldots s[y_i..x_{i+1} - 1]$ and $f_{i+1} u = u'' f_{i+1} u'$ ($u'' \in \Sigma^+$).

Considering the contraction (x_{i+1}, y_{i+1}), we have $|s| - |f_{i+1} u| + 1 = x_{i+1}$ and $|s| - |f_{i+1} u'| + 1 = y_{i+1}$ (because the contractions are in ascending order). The word s is then not yet modified after positions x_{i+1}, so $f_{i+1} u$ and $f_{i+1} u'$ are suffixes of s. Consider the state $q = State(f_{i+1})$ of $SO(s)$, according to Lemma 7:

$$State(v' f_{i+1}) = q. \tag{6}$$

The suffix $f_{i+1} u'$ of s is necessarily recognized by $SO(s)$. So the path accepting this suffix in $SO(s)$ go through the state q. Starting from q, we can read u' and

reach a final state and therefore, according to Eq. (6), $SO(s)$ also accepts the word $w_{i+1} = v' f_{i+1} u'$. Finally, the property (P) is true for all i $(0 \leq i < |\mathcal{C}|)$ and since $w_0 = s$, we can prove by induction on i that the Suffix Oracle of s recognizes all words w_i $(0 \leq i \leq |\mathcal{C}|)$. Lemma 6 allows to conclude that each suffix of words from $\mathcal{E}(s)$ is recognized by $SO(s)$.

'$\Leftarrow$': *Each word recognized by the Suffix Oracle of s is suffix of a word from $\mathcal{E}(s)$.* Let w be a word accepted by the Suffix Oracle of s and t be the longest suffix of $s = t't$ $(t, t' \in \Sigma^*)$ beginning with $w[1]$. Then a set $\mathcal{C}$ of contractions such that $t'w = Word(t't, \mathcal{C})$ exists (Lemma 9). Since the word w is suffix of $t'w$ and $t'w \in \mathcal{E}(s)$, each word accepted by $SO(s)$ is a suffix of a word from $\mathcal{E}(s)$. □

On the basis of Theorem 1 we give a similar result, which is available for Factor Oracles instead of Suffix Oracles.

Theorem 2 *Exactly all factors of words from $\mathcal{E}(s)$ are recognized by the Factor Oracle of s .*

Proof.

'$\Rightarrow$': *Each factor of words from $\mathcal{E}(s)$ is recognized by the Factor Oracle of s.*
Let $SO(s)$ be the Suffix Oracle of s, $u \in \mathcal{E}(s)$ and m a factor (*i.e.* a prefix of a suffix) of $u = mv$ $(v \in \Sigma^*)$. Then mv is accepted by $SO(s)$ (Theorem 1). Thus, a path $(e_0 \to e_{x_1} \to \ldots \to e_{x_{|mv|}})$ exists in $SO(s)$, which recognizes mv. Therefore, we conclude that m labeled a path $(e_0 \to e_{x_1} \to \ldots \to e_{x_{|m|}})$ (with $e_{x_{|m|}}$ final).

'$\Leftarrow$': *Each word recognized by the Factor Oracle of s is factor of a word from $\mathcal{E}(s)$.*
Let $SO(s)$ be the Suffix Oracle of s and m a word accepted by $FO(s)$. If $SO(s)$ recognizes m then m is a suffix of a word from $\mathcal{E}(s)$ (Theorem 1). Suppose that $SO(s)$ does not recognize m. Then $FO(s)$ recognizes m at state $e_{x_{|m|}}$ (not final in $SO(s)$). Furthermore, the path $(e_0 \to e_1 \to \ldots \to e_{|s|})$ exists in $O(s)$ and $e_{|s|}$ is final in $SO(s)$. We conclude that a path from $e_{x_{|m|}}$ to $e_{|s|}$ exists in $SO(s)$. Then, the word m is prefix of a word recognized by $SO(s)$ and therefore m is prefix of a suffix of some $u \in \mathcal{E}(s)$. Thus, m is a factor of a word of $\mathcal{E}(s)$. □

4. Properties upon Oracles & Future Works

According to Cleophas *& al.* [11], the Oracle is not minimal in number of transitions among the set of homogeneous automata. Furthermore, if we consider the set of homogeneous automata, which recognize at least all factors (resp. suffixes) of s and which have the same number of states and at most the same number of transitions than the Factor (resp. Suffix) Oracle, we show that the Oracle is not minimal on the number of accepted words. The Oracle of *axttyabcdeatzattwu* (see figure 6) has 35 transitions, the Factor Oracle accepts 247 words and the Suffix Oracle accepts 39 words. Though another homogeneous automaton (see figure 7), which recognizes at least all factors (resp. suffixes) of *axttyabcdeatzattwu* and which has only 34 transitions exists. The "Factor" version of this automaton recognizes only 236 words and its "Suffix" version accepts only 30 words. Moreover, we provide an example which has both less transitions, and less accepted words than the corresponding Oracle.

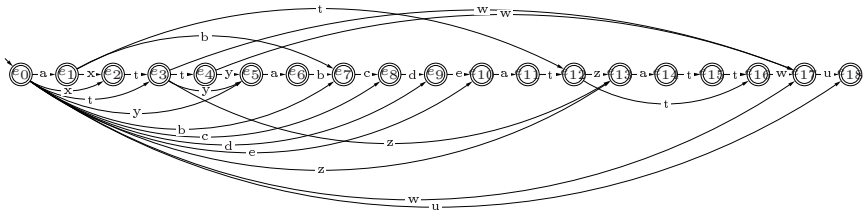

Fig. 6. Factor Oracle of the word *axttyabcdeatzattwu*.

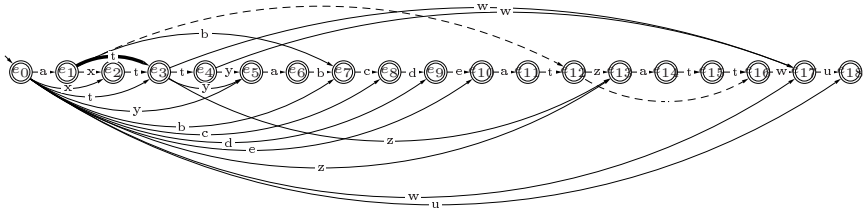

Fig. 7. This automaton (considering only the continuous lines) accepts all factors of the word *axttyabcdeatzattwu*. The bold transition (from e_1 to e_3) is the only one, which is not present in the Factor Oracle of this word (see figure 6) though the two dotted ones (from e_1 to e_{12} and from e_{12} to e_{16}) are present in the Factor Oracle, but not in this automaton.

In some cases, we observe that the number of words accepted by Oracles does not allow confidence to this structure when it is used to detect factors or suffixes of words. Even if the number of false positive can sometimes be equal to 0 (*e.g.* $aaaaaa\ldots$), it can also be exponential. Indeed, we can build a word s such that each subset of $\mathcal{C}_s^*$ is coherent and minimal. For example: $s = aabbccddee\ldots$, the set $\mathcal{C}_s^*$ of contractions, which are available on such a word, is $\{(1,2),(3,4),\ldots,(|s|-1,|s|)\}$. If we consider any non-empty subset $\mathcal{C} \subseteq (\mathcal{C}_s^* \setminus \{(1,2)\})$ of contractions, it is easy to notice that $Word(s,\mathcal{C}) \notin Fact(s)$. Besides, all words obtained from such subsets are pairwise different.

Number of subsets is: $\sum_{i=1}^{|\mathcal{C}_s^*|-1} \binom{|\mathcal{C}_s^*|-1}{i} = \sum_{i=1}^{\frac{|s|}{2}-1} \binom{\frac{|s|}{2}-1}{i} = 2^{\frac{|s|}{2}-1} - 1$. Then, the number of words, which are accepted by the Oracles and are not factor/suffix of s, is $\mathcal{O}\big(2^{|s|}\big)$.

To better use this structure, we need to improve or to slightly modify it. However, better knowledge about the Oracle structure would be useful for future works. Indeed, it could be interesting to have an empirical or a statistical estimation of the accuracy of the Oracle (time and quality of the results), when it is substituted to Tries or Suffix Trees in algorithms.

5. Acknowledgments

We would like to thank Céline Barré to have improved the English writing of this article. We also thank Irena Rusu for its useful comments and to have improved earlier versions of this article.

References

1. D. Gusfield, *Algorithms on Strings, Trees, and Sequences: computer science and computational biology* (Cambridge University Press, 1997).
2. A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen and J. Seiferas, "The smallest automaton recognizing the subwords of a text," *Theoret. Comput. Sci.* **40** (1985) 31–55.
3. E. Ukkonen, "On-line construction of suffix trees," *Algorithmica* **14** (1995) 249–260.
4. C. Allauzen, M. Crochemore and M. Raffinot, "Factor oracle, Suffix oracle," Technical Report 99–08, Institut Gaspard-Monge, Université de Marne-la-Vallée, 1999.
5. C. Allauzen, M. Crochemore and M. Raffinot, "Factor Oracle: A New Structure for Pattern Matching," *Conference on Current Trends in Theory and Practice of Informatics*, 1999, pp. 295–310.
6. C. Allauzen, M. Crochemore and M. Raffinot, "Efficient Experimental String Matching by Weak Factor Recognition," *Proc. of the 12th conference on Combinatorial Pattern Matching, Lecture Notes in Comput. Sci.* **2089** (2001) 51–72.
7. A. Lefebvre and T. Lecroq, "Computing repeated factors with a factor oracle," *Proc. of the 11th Australasian Workshop On Combinatorial Algorithms*, eds. L. Brankovic and J. Ryan (Hunter Valley, Australia, 2000) pp. 145–158.
8. A. Lefebvre and T. Lecroq, "A heuristic for computing repeats with a factor oracle: application to biological sequences," *International Journal of Computer Mathematics* **79** (2002) 1303–1315.
9. A. Lefebvre, T. Lecroq, H. Dauchel and J. Alexandre, "FORRepeats: detects repeats on entire chromosomes and between genomes," *Bioinformatics* **19** (2003) 319–326.
10. A. Lefebvre and T. Lecroq, "Compror: on-line lossless data compression with a factor oracle," *Information Processing Letters* **83** (2002) 1–6.
11. L. Cleophas, G. Zwaan and B. Watson, "Constructing Factor Oracles," *Proc. of the 3rd Prague Stringology Conference*, 2003.