Fragment Assembly through Minimal Forbidden Words*

Gabriele Fici$^{1,2}$, Filippo Mignosi$^2$, Antonio Restivo$^2$, and Marinella Sciortino$^2$

1 Institut Gaspard-Monge, Université de Marne-la-Vallée
77454 Marne-la-Vallée Cedex 2, France
fici@univ-mlv.fr

2 Dipartimento di Matematica e Applicazioni, Università di Palermo
Via Archirafi 34, 90123 Palermo, Italy
{fici,mignosi,restivo,mari}@math.unipa.it

Abstract. We give a linear-time algorithm to reconstruct a finite word $w$ over a finite alphabet $A$ of constant size starting from a finite set of factors of $w$ verifying a suitable hypothesis. We use combinatorics techniques based on the minimal forbidden words, which have been introduced in previous papers. This improves a previous algorithm which worked under the assumption of stronger hypothesis.

Keywords: Minimal forbidden word, fragment assembly, repetition index, combinatorics on words, finite automata.

Introduction

The problem of the reconstruction of a word from a set of its factors arises from several fields, as biology or cryptography. An example is the mathematical formalization of the problem of a genomic sequence reconstruction. It is known, for instance, that it is not possible to read the entire sequence of bases of a DNA molecule, but only factors of small length. The reconstruction of the original DNA sequence is complicated by other constraints, as read-errors or unknown orientation of the factors.

A theoretical simplification of the problem consists in considering a finite word as target of the reconstruction and a set of its factors as input of the problem. In general, in order to reconstruct in a unique way a word from its fragments, one has to introduce further hypothesis.

Carpi et al. [1] showed that a finite word can be uniquely reconstructed starting from a particular set of its factors. The factors needed for the reconstruction are called maximal boxes of the word.

Mignosi et al. [3, 4, 6] introduced an hypothesis of non-repetitiveness and gave two different algorithms for the sequence assembly that, under such an hypothesis, work in linear time. Such algorithms avoid one of the most common step used in solving fragment assembly problem that is the overlap phase in which every fragment is compared to each other, giving rise to a quadratic number of comparisons.

One of these algorithms is based on the notion of minimal forbidden word. Given a word $w$ over a finite alphabet $A$, a minimal forbidden word for $w$ is a finite word $v$ that is not a factor of $w$ but such that every proper factor of $v$ is a factor of $w$. The length of the longest minimal forbidden word for $w$ is noted by $m(w)$ and it is involved in the previously mentioned hypothesis of non-repetitiveness. Starting from a set $I$ of factors of $w$ containing all the factors of $w$ having length $m(w)$, it was described an algorithm able to retrieve $w$ from the set $I$ under the condition that the value of $m(w)$ is known. The authors showed that such an hypothesis on the elements of $I$ is statistically reasonable. Actually, they proved for a word $w$ randomly generated by a memoryless source with identical symbol probabilities, that the probability that $m(w)$ is $O(\log(|w|))$ converges to 1 as $|w|$ leads to infinite, so it is very likely that any factor of $w$ of length $m(w)$ is covered by at least one element of $I$.

In this paper we introduce the definition of $I$-compatibility for a finite word. Given an arbitrary finite set of finite words $I$ we say that a finite word $w$ is $I$-compatible if all the words in $I$ are factors of $w$ and if $I$ contains all the factors of $w$ having length $m(w)$. By using this definition algorithms in [3, 4, 6] work under the assumptions that there exists a word $w$ that is $I$-compatible and that $m(w)$ is known.

In this paper we improve previous result by removing the a-priori knowledge of $m(w)$, i.e. we show that the only existence of a $I$-compatible word is a sufficient condition for its unique reconstruction. Such a reconstruction can be done in linear time in the size of the set $I$.

As a second improvement, we show that it is possible to decide in linear time whether there exists a word $w$ that is $I$-compatible.

In Section 1 we recall all the needed background and we introduce some new definitions.

In Section 2 we state the Fragment Assembly Problem and we recall the techniques used in [4].

In Section 3 we show that for a given set of finite words $I$ there exists at most one $I$-compatible word.

In Section 4 we give a method that allows to retrieve the set of the minimal forbidden words for the target word $w$ starting from the input set $I$.

Finally, in Section 5 we give an algorithm for the reconstruction of the word $w$ from the set $I$ under the hypothesis that there exists a $I$-compatible word and another more general algorithm that decides whether there exists a word $w$ that is $I$-compatible and, if it exists, reconstruct it. Both the algorithms run in linear time on the size of the input set $I$. 

1 Words and automata

In this section we recall all the background we will use in the following sections. An alphabet, denoted by $A$, is a finite set of symbols. The size of $A$ is constant and it is denoted by $|A|$. A word over $A$ is a sequence of symbols from $A$. The length (or size) of a word $w$ is denoted by $|w|$. The set of all finite words over $A$ is denoted by $A^*$; the set of all the words over $A$ having a length exactly equal to $n$ is denoted by $A^n$, while the set of all the words over $A$ having a length smaller or equal to $n$ is denoted by $A^{\leq n}$. The empty word has length zero and is denoted by $\varepsilon$.

We denote by $\text{Pref}(w)$, $\text{Suff}(w)$ and $\text{Fact}(w)$ respectively the set of all prefixes, suffixes and factors of the word $w$.

Let $w$ be a word over an alphabet $A$. A finite nonempty word $v = a_0a_1 \ldots a_n$ is a minimal forbidden word for $w$ if

1. the word $v$ is not a factor of $w$,
2. the strict prefix of maximal length of $v$, $a_0a_1 \ldots a_{n-1}$, and the strict suffix of maximal length of $v$, $a_1a_2 \ldots a_n$, are factors of $w$.

We denote by $\mathcal{MF}(w)$ the set of all minimal forbidden words for $w$. By the minimality of its words we have that $\mathcal{MF}(w)$ is an anti-factorial language.

For a finite word $w$, we denote by $m(w)$ the length of the longest minimal forbidden word for $w$. One can prove (see [4]) that for a word $w$ randomly generated by a memoryless source, the parameter $m(w)$ approximates $O(\log d(n))$ where $n$ is the length of the word $w$ and $d$ is the cardinality of the alphabet.

**Remark.** The largest value that $m(w)$ can assume is $|w| + 1$, since the prefixes and the suffixes of a minimal forbidden word for a word $w$ are factors of $w$. The words $w$ having a minimal forbidden word of length $|w| + 1$ are all and the only ones of the form $w = a^n$ for a symbol $a \in A$ and a positive integer $n$. Indeed if a minimal forbidden word $u$ for $w$ has length $|w| + 1$, it must be $u = aw = wb$ for some $a, b \in A$. But in this case it is well known by the elementary theory of combinatorics on words that the only possibility is $a = b$ and $w = a^{|w|}$.

For a finite word $w$, the repetition index $r(w)$ is the length of the longest factor of $w$ that has at least two occurrences in $w$. For example the word $w = aabbbbaa$ has $r(w) = 2$.

We introduce here a new definition. A minimal forbidden word $v$ for a finite word $w$ is called a bad minimal forbidden word for $w$ if

1. the strict prefix of maximal length of $v$ appears just once as factor of $w$, and it is a suffix of $w$,
2. the strict suffix of maximal length of $v$ appears just once as factor of $w$, and it is a prefix of $w$. 

Example. Let \( w = aabbbbaa \). Then \( v' = baab \) is a bad minimal forbidden word for \( w \).

For any finite anti-factorial language \( M \), the \( L \)-automaton of \( M \) is the minimal deterministic automaton that recognizes the language \( L(M) \), and it runs in linear time on the size of \( M \). See [2] for its construction. If \( M \) is the set of the minimal forbidden words for a finite word \( w \), then the \( L \)-automaton of the language \( M \) is the minimal deterministic automaton accepting the set \( \text{Fact}(w) \) of the factors of \( w \).

Given a finite word \( w \) we can construct the set \( \mathcal{MF}(w) \) of the minimal forbidden words for \( w \) in linear time on the size of \( w \). Indeed, the algorithm \( \text{MF-trie} \), described in [2], builds the trie of the set \( \mathcal{MF}(w) \), having as input the factor automaton of \( w \), that is the minimal deterministic automaton accepting the factors of \( w \), and it runs in time \( O(|w| \times |A|) \). Moreover, the states of the trie of the set \( \mathcal{MF}(w) \) are the same as those of the factor automaton of \( w \), plus some sink states, that are the terminal states of the minimal forbidden words.

Conversely, given a finite set \( \mathcal{MF}(w) \) representing the set of the minimal forbidden words for a finite word \( w \), we can reconstruct the word \( w \) in linear time on the size of the trie representing the set \( \mathcal{MF}(w) \). The algorithm performing this operation is called \( \text{w-RECONSTRUCTION} \) and it is described in [4] and [5]. It constructs the \( L \)-automaton of the trie of the minimal forbidden words and, after deleting the sink states, it finds the longest path from the initial state in the graph of the automaton.

2 The Fragment Assembly Problem

Let \( I = \{i_1, \ldots, i_n\} \) be a set of fragments, i.e. a finite set of finite words over a given finite alphabet \( A \).

We say that a finite word \( w \) is \( I \)-compatible if

1. \( I \subseteq \text{Fact}(w) \),
2. for every \( u \in \text{Fact}(w) \) such that \( |u| \leq m(w) \) there exists at least a fragment \( i_j \in I \) such that \( u \in \text{Fact}(i_j) \).

If \( I \) is a set of factors of a finite word \( w \), we have the following definition:

**Definition 1.** A set of factors \( I \) of a finite word \( w \) is a \( k \)-cover for \( w \), for \( 0 \leq k \leq |w| \), if every factor of \( w \) of length \( k \) is a sub-factor of at least one word in \( I \). The covering index of \( I \), denoted \( C(I) \), is the largest value of \( k \) such that \( I \) is a \( k \)-cover of \( w \).

In general any set of fragments \( I \) can be a set of factors for many different words. It will have, then, a cover index for any such word.

The point 2. of the definition of \( I \)-compatibility is equivalent to the fact that the condition \( C(I) \geq m(w) \) is verified for the word \( w \) (see [4]).
The Fragment Assembly Problem is here formulated as follows:

**Fragment Assembly Problem.** Given a finite set of fragments $I$, decide whether there exists a $I$-compatible word $w$, and, if it exists, reconstruct it.

To end this section, we briefly recall the construction of the Assembly algorithm given in [4]. The inputs of the algorithm are the set $I$ and the value $m(w)$, so the algorithm works under the assumption that the value $m(w)$ is known.

Starting from the set of fragments $I = \{i_1, \ldots, i_n\}$ over the finite alphabet $A$, the first step is the construction of the concatenation word $w_1$ over the alphabet $A \cup \{\$\}$, that is the concatenation of all the strings in $I$, interspersed with the symbol $\$$, that is a special symbol not belonging to $A$, i.e. $w_1 = \$i_1\$i_2\$ \cdots \$i_n\$.

The second goal of the Assembly algorithm consists in the construction of the trie of the minimal forbidden words for $w_1$ having length smaller than or equal to $m(w)$ and not containing the symbol $. Such a construction is consequence of the following result (see [4] Proposition 5.3):

**Proposition 1.** Let $w$ be a word over a fixed alphabet $A$ and let $I$ a set of substring of $w$ such that

$$m(w) \leq C(I).$$

Then the set of minimal forbidden words for the word $w$ is exactly the set of all the minimal forbidden words for $w_1$ that do not contain the symbol $\$ and that have length smaller than or equal to $m(w)$, i.e.

$$MF(w) = MF(w_1) \cap A^{\leq m(w)}.$$

So we can retrieve the trie of the minimal forbidden words for $w$ starting from the factor automaton of $w_1$ (coming with its suffix function $h$) and the value $m(w)$. This is operation is performed in linear time $O(||I||)$ by the CREATE-TRIE algorithm, that computes the trie of the minimal forbidden words for $w_1$ and keeps those having length smaller or equal to $m(w)$ and not containing the symbol $\$.

```
CREATE-TRIE (factor automaton $F(w_1) = (Q, A \cup \{$, $i, T, \delta$), suffix function $h$, value $m(w)$)
1. for each state $p \in Q$ in breadth-first search from $i$ and each $a \in A$
2. if $\delta(p, a)$ undefined and $(p = i$ or $\delta(h(p), a)$ defined)
3. $\delta'(p, a) \leftarrow$ new sink;
4. else if $\delta(p, a) = q$ and $q$ is distant from $i$ more than $p$
5. $\delta'(p, a) \leftarrow q$;
6. In a depth-first search with respect to $\delta'$ prune all branches of the trie $T(w)$ not ending in a state that is sink and has depth smaller than or equal to $m(w)$;
7. return $T(w) = (Q', A, i', \{sinks\}, \delta')$;
```
Finally, a linear algorithm, \( w\)-RECONSTRUCTION, that reconstructs the word \( w \) from the set \( \mathcal{MF}(w) \), is used. This algorithm calls a procedure BUILDWORD that essentially finds the longest path in a DAG (Directed Acyclic Graph) by using a topological sort.

The overall ASSEMBLY algorithm is thus

\[
\text{ASSEMBLY (set of fragments } I = \{i_1, i_2, \ldots, i_n\}, \text{ value } m(w)) \\
1. \; w_1 \leftarrow i_1 i_2 \ldots i_n ; \\
2. \; \mathcal{F}(w_1) = (Q, A \cup \{$, i, T, } \leftarrow \text{FACTOR-AUTOMATON}(w_1) ; \\
3. \; \mathcal{T}(w) = (Q', A, i, \{\text{sinks}\}, } \leftarrow \text{CREATE-TRIE} (\mathcal{F}(w_1), h) ; \\
4. \; w \leftarrow w\text{-RECONSTRUCTION (} \mathcal{T}(w) ; \\
5. \; \text{return } w ; \\
\]

This algorithm runs in linear time \( O(||I||) \), where \( ||I|| \) denotes the sum of the lengths of all the strings in \( I \).

### 3 Uniqueness of the reconstruction

The main result of this section is the following theorem:

**Theorem 1.** Given a finite alphabet \( A \), and a set \( I \) of fragments over \( A \), if there exists a word \( w \) that is \( I \)-compatible, then \( w \) is unique.

We start with the following definition:

**Definition 2.** Given two finite sets of finite words \( M \) and \( M' \), we say that \( M \) is strongly included in \( M' \) if \( M \subseteq M' \) and moreover there exists at least one word \( v \in M' \) such that \( |v| > \max \{|u| : u \in M\} \). If \( M \) is strongly included in \( M' \) we note \( M \preceq M' \).

In the rest of this section we suppose that there exists a word \( w \) that is \( I \)-compatible. Let \( w' \) be another \( I \)-compatible word.

By the construction of the word \( w_1 \), concatenation of the fragments of \( I \), and by the Proposition 1, we have that

\[
- \; \mathcal{MF}(w_1) \cap A_{\leq m(w)} = \mathcal{MF}(w) \\
- \; \mathcal{MF}(w_1) \cap A_{\leq m(w')} = \mathcal{MF}(w') \\
\]

If \( m(w) = m(w') \), then \( \mathcal{MF}(w) = \mathcal{MF}(w') \), and so Fact\( (w) = \text{Fact}(w') \). Therefore \( w = w' \).

If instead \( m(w) \neq m(w') \), we have a situation in which either \( \mathcal{MF}(w) \preceq \mathcal{MF}(w') \) or \( \mathcal{MF}(w') \preceq \mathcal{MF}(w) \). The next Corollaries 1 and 2 show that this situation is impossible.

We start with a Lemma whose proof is straightforward and that will be used in the proof of the next Theorem 2.
Lemma 1. Let \( M \) be a finite anti-factorial set of words over a finite alphabet \( A \), and let \( l \) be the length of the longest word in \( M \). If a finite word \( z \) over \( A \) has the property that there exists an \( l' \geq l \) such that every factor of \( z \) of length \( l' \) does not contain any word of \( M \), then \( z \) does not contain any word of \( M \), i.e. \( z \in L(M) \).

The following theorem shows that the set of the minimal forbidden words for a finite word has a very rigid structure:

Theorem 2. Let \( w \) be a finite nonempty word over a finite alphabet \( A \), and \( X = \{ v \in MF(w) : |v| = m(w) \} \) the set of the longest minimal forbidden words for \( w \). Then the L-automaton that recognizes the language \( L = L(MF(w) \setminus X) \) of the finite words avoiding the anti-factorial language \( MF(w) \setminus X \) has some loops, so it cannot be the factor automaton of a single finite word.

Corollary 1. Let \( w \) be a finite nonempty word over a finite alphabet \( A \), and \( M = MF(w) \) the set of its minimal forbidden words. Then for every anti-factorial finite set of finite words \( M' \) such that \( M' \preceq M \), the L-automaton that recognizes the language \( L(M') \) has some loops, so it cannot be the factor automaton of a single finite word.

Corollary 2. Let \( M = MF(w) \) be the (anti-factorial) set of the minimal forbidden words for a finite nonempty word \( w \) over a finite alphabet \( A \). Then for every anti-factorial finite set of finite words \( M' \) such that \( M \preceq M' \), the L-automaton that recognizes the language \( L(M') \) cannot be the factor automaton of a single finite word.

4 Finding the minimal forbidden words for \( w \)

We now suppose to have a set of fragments \( I \) and that there exists a (unique) \( I \)-compatible word \( w \). So we know that there exists a finite word \( w \) such that the fragments of \( I \) are factors of \( w \) and the condition \( C(I) \geq m(w) \) is verified, and we want to reconstruct the word \( w \).

In particular, we are interested in finding the minimal forbidden words for the word \( w \). This will allow us to reconstruct the word \( w \) using the \( w \)-RECONSTRUCTION procedure.

In this section we find a way to deduce the set \( MF(w) \) from the set \( MF(w_1) \) without the explicit knowledge of the value \( m(w) \).

The following two Propositions are given in [4] without proof (Remarks 3.3 and 5.4).

Proposition 2. Let \( w \) be a finite word over a finite alphabet \( A \). Then
\[
r(w) = m(w) - 2
\]
where \( r(w) \) is the repetition index of \( w \) and \( m(w) \) is the length of the longest minimal forbidden word for \( w \).
We focus now on the structure of the set $$(\mathcal{MF}(w_1) \cap A^*) \setminus \mathcal{MF}(w)$$, i.e. of the minimal forbidden words for $w_1$ not containing the symbol $\$" that are not minimal forbidden words for $\mathcal{MF}(w)$.

**Remark.** Since by the Proposition 1 we have that $\mathcal{MF}(w_1) \cap A^{\leq m(w)} = \mathcal{MF}(w)$, every word belonging to $$(\mathcal{MF}(w_1) \cap A^*) \setminus \mathcal{MF}(w)$$ has a length greater than $m(w)$.

Let $\mathcal{S}$ be the set of the minimal forbidden words for $w_1$ not containing the symbol $\$$, of the form $v = au\$$, such that

- the words $au\$$ and $\$$ub$ are factors of $w_1$,
- the words $aux$ and $xub$ are not factors of $w_1$, for every $x \in A$.

We will show that the knowledge of this set allows us to retrieve the set $\mathcal{MF}(w)$ from the set $\mathcal{MF}(w_1)$.

Note that the set $\mathcal{S}$ can also be empty.

The second Proposition is the following:

**Proposition 3.** The elements of $\mathcal{MF}(w_1) \cap A^*$ that do not belong to $\mathcal{MF}(w)$ (i.e. having a length greater than $m(w)$) are of the form $avb$, where $av\$$ and $\$$vb$ are factors of the word $w_1$, and in the word $w_1$ the factor $av$ can only be followed by the symbol $\$$ and the factor $vb$ can only be preceded by the symbol $\$$. In other words $$(\mathcal{MF}(w_1) \cap A^*) \setminus \mathcal{MF}(w) \subseteq \mathcal{S}.$$ Set

$$S = (\mathcal{MF}(w_1) \cap A^*) \setminus \mathcal{S}.$$ We want to give a condition under which $S = \mathcal{MF}(w)$.

Remind that, starting with the set $I$, we do not know anything about the existence of bad minimal forbidden words for the word $w$.

**Proposition 4.** If $w$ is $I$-compatible, then every minimal forbidden word for $w$ that is not a bad one does not belong to $\mathcal{S}$.

**Theorem 3.** If $w$ is $I$-compatible, and if no bad minimal forbidden word for $w$ exists, then $S = \mathcal{MF}(w)$.

Thus, if $w$ is $I$-compatible and no bad minimal forbidden words for $w$ exists, we have $\mathcal{MF}(w) = (\mathcal{MF}(w_1) \cap A^*) \setminus \mathcal{S}$, so we can deduce the set $\mathcal{MF}(w)$ from the set $\mathcal{MF}(w_1)$.

Now it rests to consider the case of the existence of bad minimal forbidden words for $w$.

**Proposition 5.** For a finite word $w$ over a finite alphabet $A$ it can exist at most one bad minimal forbidden word.
Proposition 6. If \( w \) is \( I \)-compatible and \( v \) is a bad minimal forbidden word for \( w \), then \( v \in \overline{S} \).

So we have the following theorem:

Theorem 4. If \( w \) is \( I \)-compatible, then the elements of the set \( \overline{S} \) are the minimal forbidden words for \( w_1 \) not containing the symbol \( \$ \) that are not minimal forbidden words for \( w \) and eventually the bad minimal forbidden word for \( w \).

Suppose that there exists a bad minimal forbidden word for \( w \). Let \( l_1 \) be the length of the shortest word in \( \overline{S} \). If a word \( v \) belongs to \( \overline{S} \), then by the Theorem 4, either \( v \) is the bad minimal forbidden word for \( w \), and in this case it has length smaller or equal to \( m(w) \), or \( v \) belongs to \( (MF(w_1) \cap A^*) \setminus MF(w) \), and in this case we know that it must have length greater than \( m(w) \).

So, if there exists a bad minimal forbidden word for \( w \), then \( \overline{S} \) contains only one word of minimal length \( l_1 \), that is the bad minimal forbidden word for \( w \). Thus, if we set

\[
\overline{S} = S \setminus \{ v \in S : |v| = l_1 \},
\]

we have \( MF(w) = (MF(w_1) \cap A^*) \setminus \overline{S} = MF(w_1) \cap A^{\leq l_2} \), where \( l_2 \geq l_1 \) is the length of the second shortest word in \( S \) (with the convention that \( l_2 = l_1 \) if and only if there are in \( S \) more than one word having minimal length), or infinity if \( S \) contains just one word.

Example. The case \( m(w) = |w| + 1 \) is a very special case, as we have seen in a previous Remark, and the words verifying this property are of the form \( w = a^n \), for a letter \( a \in A \) and a positive integer \( n \). In this case \( v = a^{n+1} \) is a minimal forbidden word for \( w \) having length \( |w| + 1 \). Note that any set \( I \) of fragments for which the word \( w \) is \( I \)-compatible must contain the whole word \( w \), and eventually some other powers \( a^m \), with \( m \leq n \). In this case we have \( MF(w_1) \cap A^* = \{ a^{n+1} \} \cup \{ A \{ a \} \} \), \( \overline{S} = \{ a^{n+1} \} \), and \( \overline{S} = \emptyset \).

Therefore \( MF(w) = MF(w_1) \cap A^* \setminus \overline{S} = \{ a^{n+1} \} \cup \{ A \{ a \} \} \).

5 A new algorithm for the fragment assembly problem

We start with the construction of a procedure that computes in linear time \( O(|A|^2 \times ||I||) \) the values \( l_1 \) and \( l_2 \), representing respectively the length of the shortest and of the second shortest element of \( \overline{S} \), if they exist, starting from the factor automaton and the trie of the minimal forbidden words for the word \( w_1 \).

The first algorithm is the \( \overline{S} \)-construction. It labels the sink states of the trie of the minimal forbidden words for \( w_1 \).

The algorithm uses a FIFO (First In First Out) file \( F \), of which the entries are couples of states, the first one corresponding to a breadth-first-search on the factor
automaton of $w_1$, and the second one corresponding to a breadth-first-search on the trie of the minimal forbidden words for $w_1$.

The behavior of the algorithm is the following. First it fixes a symbol $x \in A \cup \{\$\}$. Then it fixes a letter $a \in A$ and starts a breadth-first-searches on the trie of the minimal forbidden words for the word $w_1$. At the end of the exploration of the trie it labels with the symbol $x$ the sink states corresponding to the minimal forbidden words $v = aub$ verifying:

1. $v$ does not contain the symbol $\$
2. $xub$ is a factor of $w_1$
3. in the word $w_1$ the factor $au$ can only be followed by the symbol $\$

To perform this operation, the algorithm does at the same time a breadth-first-search on the factor automaton of $w_1$, to ensure that the factor $xub$ is indeed a factor of $w_1$.

At the end of the procedure, the sink states labelled only by the symbol $\$ are the states corresponding to the words of $\overline{S}$.

In the worst theoretical case, if we note $k = |A|$, the algorithm does $k(k + 1)$ breadth-first-searches ($k + 1$ possibilities for the letter $x$ at line 1., and $k$ for the letter $a$ at line 2.) on the trie $\mathcal{MF}(w_1)$. Since the set of the states that are not sink of the trie $\mathcal{MF}(w_1)$ is a subset of the set of the states of the factor automaton $\mathcal{F}(w_1)$, and since a breadth first search is a linear standard procedure, the algorithm is linear on $||I||$, where $||I||$ denotes the sum of the lengths of all the strings in $I$.

The second algorithm, $l_1, l_2$-FINDING, does a breadth-first-search on the labelled trie $T'(w_1)$ using a FIFO file $F$ having two entries: the first one is a state and second one is the distance of this state from the initial state. The algorithm returns the lengths $l_1$ and $l_2$ respectively of the shortest and of the second shortest minimal forbidden words for $w_1$ whose label in the trie $T'(w_1)$ is only the symbol $\$, i.e. the shortest and the second shortest elements of $\overline{S}$. If $\overline{S}$ is empty, then the algorithm sets both $l_1$ and $l_2$ equals to infinity. If $\overline{S}$ contains just one element, then the algorithm sets $l_2$ equal to infinity. Its linear time complexity follows from the linear time complexity of the standard breadth-first-search procedure on a finite graph.

Now, we describe the Fragment Assembly 1 algorithm. It reconstructs the word $w$ from a set of fragments $I$ under the hypothesis that there exists a $I$-compatible word $w$.

The first step of the procedure is the construction of the concatenation word $w_1$, that can be easily done in linear time $O(||I||)$.

Then we can construct the factor automaton of $w_1$. Remember that the factor automaton of a word $v$ over the alphabet $A$ can be computed in linear time $O(|v| \times |A|)$ and has no more than $2|v|$ states, see for instance [2].

Now, we can construct in linear time $O(|w_1|) = O(||I||)$ the trie $T(w_1)$ of the set $\mathcal{MF}(w_1)$, by using the MF-Trie algorithm.
Once we have constructed both the factor automaton and the trie of the minimal forbidden words for \(w_1\), we can use the two algorithms \(S\)-construction and \(l_1, l_2\)-finding to find the values \(l_1\) and \(l_2\).

Now we apply the CREATE-Trie algorithm with the value \(l_1 - 1\) instead of the value \(m(w)\) at the line 6, and we obtain the trie \(T(l_1 - 1)\) that represents the set \(M_1 = MF(w_1) \cap A^{\leq l_1 - 1}\) (we use the convention that \(A^{\leq \infty} = A^*\)). If no bad minimal forbidden word exists for the \(I\)-compatible word \(w\), then \(M_1 = S = MF(w)\), so we can easily reconstruct \(w\) by using the \(w\)-RECONSTRUCTION procedure applied to the trie \(T(l_1 - 1)\).

If instead there exists a bad minimal forbidden word for the \(I\)-compatible word \(w\), then the \(L\)-automaton applied to the trie \(T(l_1 - 1)\) will give, by the Corollary 1, an automaton \(F(l_1 - 1)\) with some loops. Note that checking whether a finite directed graph contains loops or not is a standard linear procedure (by using for example a depth-first-search).

So, if \(F(l_1 - 1)\) contains some loops, and since by the hypothesis there exists a \(I\)-compatible word \(w\), then we can state that there exists a bad minimal forbidden word for \(w\), and so, by the Theorem 4, the CREATE-Trie algorithm, with the value \(l_2 - 1\) instead of the value \(m(w)\) at line 6, will produce the trie of the minimal forbidden words for the word \(w\), that so can be reconstructed by using the \(w\)-RECONSTRUCTION procedure applied to the trie \(T(l_2 - 1)\).

We are now quite close to the solution of the Fragment Assembly Problem as formulated in Section 2. The last step consists in eliminating the hypothesis on the existence of a \(I\)-compatible word.

So we start only with an arbitrary set \(I\) of finite words over a finite alphabet \(A\).

The following Fragment Assembly 2 algorithm completely answers to the Fragment Assembly Problem, and it produces its output in linear time \(O(||I||)\).

The first steps of the algorithm are the same as those of the Fragment Assembly 1 algorithm. Once we have constructed the automaton \(F(l_1 - 1)\) we have to check whether it is (after deleting the sink states) the factor automaton of a \(I\)-compatible word.

If it contains loops, clearly it cannot be the factor automaton of a single finite word.

If it does not contain loops, how can we decide whether it is the factor automaton of a \(I\)-compatible word? First, the factor automaton of a single finite word always contains a unique longest path from the initial state, and it is the path corresponding to the longest factor of the word, that is the word itself. So if \(F(l_1 - 1)\) contains two or more paths of maximal length (one can check this in linear time by using a simple adaptation of the topological sort procedure on a directed acyclic graph), we can state that no \(I\)-compatible word exists; otherwise, by the Corollary

3 In the general construction the \(L\)-automaton has the same states of its input trie, but we always suppose to delete the sink states of the \(L\)-automaton after its construction. So we do not consider loops on the sink states.
1, \( \mathcal{F}(l_1 - 1) \) should contain loops. If instead \( \mathcal{F}(l_1 - 1) \) contains just one path of maximal length, set \( w \) this path. Now, one can construct in linear time on the size of \( w \) (and so on \( ||I|| \)) the factor automaton of \( w \), noted by \( \mathcal{F}(w) \). We can now compare the automata \( \mathcal{F}(w) \) and \( \mathcal{F}(l_1 - 1) \) (it is well known that checking the equality between two finite deterministic automata can be done in linear time).

If \( \mathcal{F}(w) \neq \mathcal{F}(l_1 - 1) \), then we can state that no \( I \)-compatible word exists, if not we should retrieve its factor automaton with \( \mathcal{F}(l_1 - 1) \) (or \( \mathcal{F}(l_2 - 1) \) if \( \mathcal{F}(l_1 - 1) \) contains loops, that is not the case here).

If \( \mathcal{F}(w) = \mathcal{F}(l_1 - 1) \) then we have found a \( I \)-compatible word, \( w \), as the following Theorem shows.

**Theorem 5.** If \( \mathcal{F}(l_1 - 1) \), the automaton that recognizes the set \( L(\mathcal{MF}(w_1) \cap A \leq l_1 - 1) \), is the factor automaton of a finite word \( w \), then \( w \) is a \( I \)-compatible word.

The previous Theorem also shows that our condition is tight, in the sense that the algorithm **Fragment Assembly 2** cannot retrieve a finite word \( w \) that is not \( I \)-compatible.

If \( \mathcal{F}(l_1 - 1) \) contains loops, we can try with the automaton \( \mathcal{F}(l_2 - 1) \), obtained from the Create-Trie procedure with the value \( l_2 \).

If it contains loops, we can state that no \( I \)-compatible word exists, otherwise we should obtain an automaton without loops, as in the **Fragment Assembly 1** algorithm. If instead it does not contain loops, we can apply the same test as that we did for \( \mathcal{F}(l_1 - 1) \).

The linear time complexity of the whole **Fragment Assembly 2** algorithm follows from the linear time complexity of all the used procedures.

**References**