Minimal Forbidden Words and Factor Automata

M. Crochemore[⋆,1], F. Mignosi[2], and A. Restivo[2]

[1] Institut Gaspard-Monge mac@univ-mlv.fr
[2] Università di Palermo [mignosi,restivo]@altair.math.unipa.it

Abstract. Let $L(M)$ be the (factorial) language avoiding a given anti-factorial language M. We design an automaton accepting $L(M)$ and built from the language M. The construction is effective if M is finite.
If M is the set of minimal forbidden words of a single word v, the automaton turns out to be the factor automaton of v (the minimal automaton accepting the set of factors of v).
We also give an algorithm that builds the trie of M from the factor automaton of a single word. It yields a non-trivial upper bound on the number of minimal forbidden words of a word.

Keywords: factorial language, anti-factorial language, factor code, factor automaton, forbidden word, avoiding a word, failure function.

1 Introduction

Let $L \subseteq A^*$ be a *factorial* language, *i.e.*, a language containing all factors of its words. A word $w \in A^*$ is called a *minimal forbidden word* for L if $w \notin L$ and all proper factors of w belong to L. We denote by $MF(L)$ the language of minimal forbidden words for L.

The study of combinatorial properties of $MF(L)$ helps investigate the structure of the language L or of the system it describes. For instance, locally testable factorial languages (cf [8]) are characterized by the fact that the corresponding languages of minimal forbidden words are finite. In the context of Symbolic Dynamics they correspond to systems of finite type.

Another example is given by a language L that is the set of factors of an infinite word: in this case, as shown in [2], the elements of $MF(L)$ are closely related to the *bispecial* factors (cf. [6], [7] and [3]) of the infinite word.

A measure of complexity of the language L is introduced in [2] based on the function F_L, that counts, for any n, the number of words of length n in $MF(L)$. Authors prove that the growth of $F_L(n)$ as well as the topological entropy of $MF(L)$ are topological invariants of the dynamical system defined by L. This result provides a usefull tool to show that some systems are not isomorphic, which comes in addition to other notion like the ordinary notion of entropy and the zeta function, for example.

Finally, [5] considers properties of languages defined by finite forbidden sets of words. Authors define the Möbius function for these languages.

⋆ Work by this author is supported in part by Programme "Génomes" of C.N.R.S.

In this paper we focus on the transformations between L and $MF(L)$. We first design an automaton accepting $L(M)$ and that is built from the language M. When M is a finite set the transformation is effective. Moreover, if M is given by its digital tree, that is, its tree-like deterministic automaton, the algorithm is very similar to the algorithm of Aho and Corasick that builds a pattern-matching machine for a finite set of words [1].

In a second part we consider the particular situation of a language that is the set of factors of a single word v. The construction of its *factor automaton*, the minimal deterministic automaton accepting the factors of v (see [4]) is known to be rather intricate. It is remarkable that the preceding transformation yields exactly the factor automaton of v when the input if the set M of minimal forbidden words of v. We also give an algorithm that realizes the converse transformation, building the trie of M from the factor automaton of v. A corollary of the algorithm is a non-trivial upper bound on the number of minimal forbidden words of a word.

The complexities of algorithms described in this paper are all linear in the size of their input or output. Therefore, the design of possible faster algorithms relies on different representations of objects, which is not the aim of the paper.

2 Avoiding an anti-factorial language

Let A be a finite alphabet and A^* be the set of finite words drawn from the alphabet A, the empty word ϵ included. Let $L \subseteq A^*$ be a *factorial language*, *i.e.* a language satisfying: $\forall u, v \in A^* \ uv \in L \Longrightarrow u, v \in L$. The complement language $L^c = A^* \setminus L$ is a (two-sided) ideal of A^*. Denote by $MF(L)$ the base of this ideal, we have $L^c = A^* MF(L) A^*$.

The set $MF(L)$ is called the set of *minimal forbidden words* for L. A word $v \in A^*$ is forbidden for the factorial language L if $v \notin L$, which is equivalent to say that v occurs in no word of L. In addition, v is minimal if it has no proper factor that is forbidden.

One can note that the set $MF(L)$ uniquely characterizes L, just because

$$L = A^* \setminus A^* MF(L) A^*. \tag{1}$$

The following simple observation provides a basic characterization of minimal forbidden words.

Remark 1 *A word $v = a_1 a_2 \cdots a_n$ belongs to $MF(L)$ iff the two conditions hold:*

- *v is forbidden, (i.e., $v \notin L$),*
- *both $a_1 a_2 \cdots a_{n-1} \in L$ and $a_2 a_3 \cdots a_n \in L$ (the prefix and the suffix of v of length $n-1$ belong to L).*

The remark translates into the equality:

$$MF(L) = AL \cap LA \cap (A^* \setminus L). \tag{2}$$

As a consequence of both equalities (1) and (2) we get the following proposition.

Proposition 1 *For a factorial language L, languages L and $MF(L)$ are simultaneously rational, that is, $L \in Rat(A^*)$ iff $MF(L) \in Rat(A^*)$.*

The set $MF(L)$ is an *anti-factorial language* or a *factor code*, which means that it satisfies: $\forall u, v \in MF(L)\ u \neq v \Longrightarrow u$ is not a factor of v, property that comes from the minimality of words of $MF(L)$.

We introduce a few more definitions.

Definition 1 *A word $v \in A^*$* avoids the set *M, $M \subseteq A^*$, if no word of M is a factor of v, (i.e., if $v \notin A^*MA^*$). A language L* avoids *M if every words of L avoid M.*

From the definition of $MF(L)$, it readily comes that L is the largest (according to the subset relation) factorial language that avoids $MF(L)$. This shows that for any anti-factorial language M there exists a unique factorial language $L(M)$ for which $M = MF(L)$. The next remark summarizes the relation between factorial and anti-factorial languages.

Remark 2 *There is a one-to-one correspondence between factorial and anti-factorial languages. If L and M are factorial and anti-factorial languages respectively, both equalities hold: $MF(L(M)) = M$ and $L(MF(L)) = L$.*

We also refer to the next definition that is to be considered in the context of dynamical systems (see [9] for example).

Definition 2 *The factorial language L is said to be* of finite type *when $MF(L)$ is finite.*

Finally, with an anti-factorial finite language M we associate the finite automaton $\mathcal{A}(M)$ as described below. The automaton is deterministic and complete, and, as shown at the end of the section by Theorem 3, the automaton accepts the language $L(M)$.

The automaton $\mathcal{A}(M)$ is the tuple (Q, A, i, T, F) where

- the set Q of states is $\{w \mid w$ is a prefix of a word in $M\}$,
- A is the current alphabet,
- the initial state i is the empty word ϵ,
- the set T of terminal states is $Q \setminus M$.

States of $\mathcal{A}(M)$ that are words of M are sink states. The set F of transitions is partitioned into the three (pairwise disjoint) sets F_1, F_2, and F_3 defined by:

- $F_1 = \{(u, a, ua) \mid ua \in Q, a \in A\}$ (forward edges or tree edges),
- $F_2 = \{(u, a, v) \mid u \in Q \setminus M, a \in A, ua \notin Q, v$ longest suffix of ua in $Q\}$ (backward edges),
- $F_3 = \{(u, a, u) \mid u \in M, a \in A\}$ (loops on sink states).

The transition function defined by the set F of arcs of $\mathcal{A}(M)$ is noted δ.

Remark 3 *One can easily prove from definitions that*

1. *if $q \in Q \setminus (M \cup \{\epsilon\})$, all transitions arriving on state q are labeled by the same letter $a \in A$,*
2. *from any state $q \in Q$ we can reach a sink state, i.e., q can be extended to a word of M.*

Definition 3 *For any $v \in A^*$, q_v denotes the state $\delta(\epsilon, v)$, target of the unique path in $\mathcal{A}(M)$ starting at the initial state and labeled by v.*

Since $\mathcal{A}(M)$ is a complete automaton, q_v is always defined. In the automaton $\mathcal{A}(M)$ states are words, but to avoid misunderstandings we sometimes write "the word corresponding to q_v" instead of just "the word q_v".

Remark 4 *Note that if v is a state of $\mathcal{A}(M)$ we have $q_v = v$.*

We are now ready to state the next lemma (which proof is by induction on v) that is used in the proof of Theorem 3, the main result of the section.

Lemma 2 *Let M be an anti-factorial language and consider $\mathcal{A}(M)$. Let $v \in A^*$ be such that, for any proper prefix u of v, q_u is not a sink state ($q_u \notin M$). Then,*

1. *the word q_v is a suffix of v,*
2. *q_v is the longest suffix of v that is also a state of $\mathcal{A}(M)$ (or $\forall q \in Q$ q suffix of $v \Longrightarrow q$ suffix of q_v).*

Proof. By induction on $|v|$. $\bowtie$

Denoting by $Lang(\mathcal{A})$ the language accepted by an automaton $\mathcal{A}$, we get the main theorem of the section.

Theorem 3 *For any anti-factorial language M, $Lang(\mathcal{A}(M)) = L(M)$.*

Proof. We first prove $L(M) \subseteq Lang(\mathcal{A}(M))$. We have to show that if v is a word that avoids M then $v \in Lang(\mathcal{A}(M))$. Assume *ab absurdo* that $v \notin Lang(\mathcal{A}(M))$; therefore q_v is a sink state. Let u be the shortest prefix of v for which q_u is a sink state (note that $q_u = q_v$). By lemma 2 statement 1, q_u is a suffix of u, but q_v is by definition an element of M, and so v does not avoid M, a contradiction.

We then prove $Lang(\mathcal{A}(M)) \subseteq L(M)$. Let $v \in Lang(\mathcal{A}(M))$. Let us suppose *ab absurdo* that v does not avoid M, *i.e.*, $v = uwz$ for some $w \in M, u, z \in A^*$. We choose uw as the shortest prefix of v that belongs to A^*M. Since $w \in M$ it is by definition a state of $\mathcal{A}(M)$; since w is a state that is a suffix of uw, by Lemma 2 statement 2, w is a suffix of q_{uw}. But q_{uw}, which is by definition a state of $\mathcal{A}(M)$, is a prefix of an element w' of M. Since w is a suffix of a prefix of w', w is a factor of w', a contradiction because M is anti-factorial. $\bowtie$

The above definition of $\mathcal{A}(M)$ turns into the algorithm below, called L-AUTOMATON, that builds the automaton from a finite anti-factorial set of words. The input is the trie $\mathcal{T}$ that represents M. It is a tree-like automaton accepting the set M and, as such, it is noted (Q, A, i, T, δ'). The procedure can be adapted to test whether $\mathcal{T}$ represents an anti-factorial set, or even to generate the trie of the anti-factorial language associated with a set of words.

In view of Equality 1, the design of the algorithm remains to adapt the construction of a pattern matching machine (see [1] or [4]) The algorithm uses a function f called a *failure function* and defined on states of $\mathcal{T}$ as follows. States of the trie $\mathcal{T}$ are identified with the prefixes of words in M. For a state au ($a \in A$, $u \in A^*$), $f(au)$ is $\delta'(i, u)$, quantity that may happen to be u itself. Note that $f(i)$ is undefined, which justifies a specific treatment of the initial state in the algorithm.

L-AUTOMATON (trie $\mathcal{T} = (Q, A, i, T, \delta')$)
for each $a \in A$
    if $\delta'(i, a)$ defined
        set $\delta(i, a) = \delta'(i, a)$;
        set $f(\delta(i, a)) = i$;
    else
        set $\delta(i, a) = i$;
for each state $p \in Q \setminus \{i\}$ in width-first search **and** each $a \in A$
    if $\delta'(p, a)$ defined
        set $\delta(p, a) = \delta'(p, a)$;
        set $f(\delta(p, a)) = \delta(f(p), a)$;
    else if $p \notin T$
        set $\delta(p, a) = \delta(f(p), a)$;
    else
        set $\delta(p, a) = p$;
return $(Q, A, i, Q \setminus T, \delta)$;

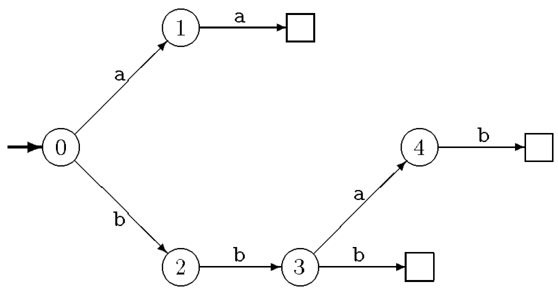

Fig. 1. Trie of the factor code $\{\mathtt{aa}, \mathtt{bbaa}, \mathtt{bbb}\}$ on the alphabet $\{\mathtt{a}, \mathtt{b}\}$. Squares represent terminal states.

Example. Figure 1 displays the trie that accepts $M = \{\mathtt{aa}, \mathtt{bbaa}, \mathtt{bbb}\}$. It is an anti-factorial language. The automaton produced from the trie by algorithm L-AUTOMATON is shown in Figure 2. It accepts the prefixes of $(\mathtt{ab} \cup \mathtt{b})(\mathtt{ab})^*\mathtt{ba}$ that are all the words avoiding M.

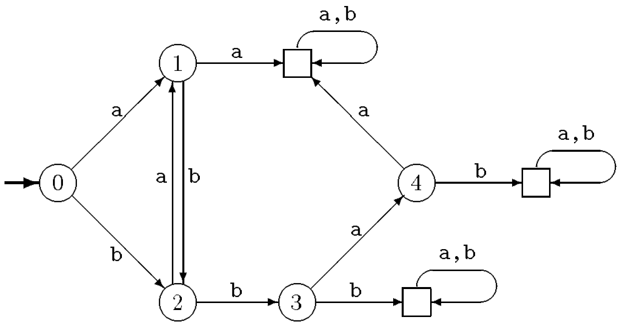

Fig. 2. Automaton accepting the words that avoid the set $\{\mathtt{aa}, \mathtt{bbaa}, \mathtt{bbb}\}$. Squares represent non-terminal states (sink states).

Theorem 4 *Let $\mathcal{T}$ be the trie of an anti-factorial language M. Algorithm* L-AUTOMATON *builds a complete deterministic automaton accepting $L(M)$.*

Proof. The automaton produced by the algorithm has the same set of states as the input trie. It is clear that the automaton is deterministic and complete.

Let $u \in A^+$ and $p = \delta(i, u)$. A simple induction on $|u|$ shows that the word corresponding to $f(p)$ is the longest proper suffix of u that is a prefix of some word in M. This notion comes up in the definition of the set of transitions F_2 in the automaton $\mathcal{A}(M)$. Therefore, the rest of the proof just remains to check that instructions implement the definition of $\mathcal{A}(M)$. $\bowtie$

Theorem 5 *Algorithm* L-AUTOMATON *runs in time $O(|Q| \times |A|)$ on input $\mathcal{T} = (Q, A, i, T, \delta')$ if transition functions are implemented by transition matrices.*

3 Factor automaton of a single word

In this section we specialize the previous results to the language of factors of a single word. It is proved below that the contruction of Section 2 yields the factor automaton (minimal dterministic automaton accepting the factors) of the word (see Theorem 7). The minimality of the automaton seems to be exceptional because, for example, the same construction applied to the set $\{aa, ab\}$ does not provide a minimal automaton.

The reverse construction that produces the trie of minimal forbidden words from the factor automaton is described in the next section.

We consider a fixed word $v \in A^*$ and denote by $\mathcal{F}(v)$ be the language of factors of v.

Proposition 6 *The language $\mathcal{F}(v)$ is of finite type.*

Proof. Indeed, factors of v, of lengths less than $|v|+1$, avoid all words of length exactly $|v|+1$. Therefore, every minimal forbidden word of $\mathcal{F}(v)$ has length at most $|v|+1$. $\bowtie$

The result of the previous proposition is made more precise in the next section, but an immediate consequence of it and of the definition of the automaton $\mathcal{A}(M)$ for an anti-factorial language M, the automaton $\mathcal{A}(MF(\mathcal{F}(v)))$ has a finite number of states. The next statement gives a complete characterization of the automaton as the factor automaton of v.

Theorem 7 *For any $v \in A^*$, the automaton obtained from $\mathcal{A}(MF(\mathcal{F}(v)))$ by removing its sink states is the minimal deterministic finite automaton accepting the language $\mathcal{F}(v)$ of factors of v.*

Proof. The automaton $\mathcal{A}(MF(\mathcal{F}(v)))$ is already a deterministic finite automaton that accepts the language $\mathcal{F}(v)$ by Theorem 3. We only have to prove that it is minimal after removing the sink states.

Suppose *ab absurdo* that there exist two equivalent non-sink states p, q in Q. By the standard equivalence relation of undistinginshability and by construction $p, q \in \mathcal{F}(v)$. Hence, $v = xpy$ and $v = x'qy'$ and we can choose x and x' of minimal length. We consider two cases:

(i) $|xp| \neq |x'q|$,
(ii) $|xp| = |x'q|$.

Case (i). We can suppose for example that $|xp| < |x'q|$ (the case $|xp| > |x'q|$ is handled symmetrically). Then, $xpy \in \mathcal{F}(v)$ implies that $\delta(p, y)$ is not a sink state, hence, by the equivalence $\delta(q, y)$ is not a sink state, that is, $qy \in \mathcal{F}(v)$ by Remark 4. Therefore, $v = x"qyz$ where $|x"| \geq |x'|$ by the choice of x' (of minimal length). Hence, $|v| \geq |x'| + |q| + |y| + |z| > |xp| + |y| = |v|$, a contradiction.
Case (ii). The equality $|xp| = |x'q|$ implies either that p is a suffix of q or the converse. Let us suppose for example that $p = sq$ for some word $s \neq \epsilon$. By Remark 3 statement 2, there exists $w = pz$ that belongs to $MF(\mathcal{F}(v))$. By the equivalence, qz is also a sink state and, again by the equivalence, for no proper prefix u of qz, q_u is a sink state. Hence, by Lemma 2.1, q_{qz} is an element of $MF(\mathcal{F}(v))$, that is, a suffix of qz. Since $p = sq, s \neq \epsilon$, q_{qz} is a proper suffix of pz against the anti-factorial property of $MF(\mathcal{F}(v))$. A contradiction again.

After cases (i) and (ii) it appears that there cannot exist two different non-sink states p, q in Q that are equivalent. Therefore the automaton without sink states is minimal, which ends the proof. $\bowtie$

4 Minimal forbidden words of a word

We end the article by an algorithm that builds the trie accepting the language $MF(\mathcal{F}(v))$ of minimal words avoided by v. This is an implementation of the inverse of the transformation described in Section 2. Its design follows Equality 2. A corollary of the transformation gives a bound on the number of minimal forbidden words of a single word, which improves on the bound coming readily from Proposition 6.

MF-TRIE (factor automaton $\mathcal{A} = (Q, A, i, T, \delta)$ and its suffix function s)

```
for each state p ∈ Q in width-first search from i and each a ∈ A
    if δ(p, a) undefined and (p = i or δ(s(p), a) defined
        set δ'(p, a) = new sink;
    else if δ(p, a) = q and q not already treated
        set δ'(p, a) = q;
return (Q, A, i, {sinks}, δ');
```

The input of algorithm MF-TRIE is the factor automaton of word v. It includes the failure function defined on the states of the automaton and called s. This function is a by-product of efficient algorithms that build the factor automaton (see [4]). It is defined as follows. Let $u \in A^+$ and $p = \delta(i, u)$. Then, $s(p) = \delta(i, u')$ where u' is the longest suffix of u for which $\delta(i, u) \neq \delta(i, u')$. It can be shown that the definition of $s(p)$ does not depend on the choice of u.

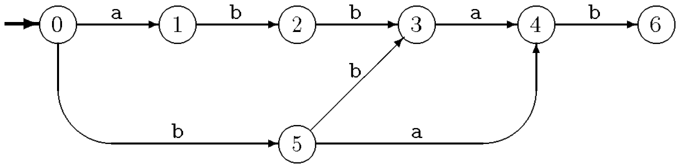

Fig. 3. Factor automaton of abbab; all states are terminal.

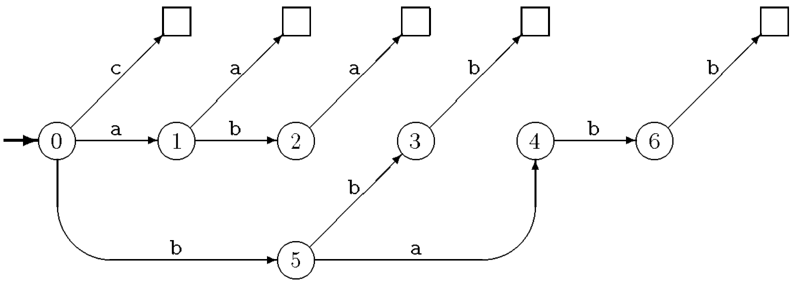

Fig. 4. Trie of minimal forbidden words of $\mathcal{F}$(abbab) on the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Squares represent terminal states.

Example Consider the word $v = \mathtt{abbab}$ on the alphabet $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$. Its factor automaton is displayed in Figure 3. The failure function s defined on states has values: $s(1) = s(5) = 0$, $s(2) = s(3) = 5$, $s(4) = 1$, $s(6) = 2$. Algorithm MF-TRIE produces the trie of Figure 4 that represents the set of five words $\{\mathsf{aa}, \mathsf{aba}, \mathsf{babb}, \mathsf{bbb}, \mathsf{c}\}$.

Theorem 8 *Let $\mathcal{A}$ be the factor automaton of a word $v \in A^*$. (It accepts the language $\mathcal{F}(v)$.) Algorithm* MF-TRIE *builds the tree-like deterministic automaton accepting $MF(\mathcal{F}(v))$ the set of minimal forbidden words of $\mathcal{F}(v)$.*

Corollary 9 *A word $v \in A^*$ has no more than $2(|v|-2)(|A_v|-1)+|A|$ minimal forbidden words if $|v| \geq 3$, where A_v is the set of letters occurring in v. The bound becomes $|A|+1$ if $|v| < 3$.*

Proof. The number of words in $MF(\mathcal{F}(v))$ is the number of sink states created during the execution of algorithm MF-TRIE. These states have exactly one ingoing arc originated at a state of the factor automaton $\mathcal{A}$ of v. So, we have to count these arcs.

From the initial state of $\mathcal{A}$ there is exactly $|A| - |A_v|$ such arcs. From the (unique) state of $\mathcal{A}$ without outgoing arc, there are at most $|A_v|$ such arcs. From other states there are at most $|A_v| - 1$ such arcs.

For $|v| \geq 3$, it is known that $\mathcal{A}$ has at most $2|v|-2$ states (see [4]). Therefore, $|MF(\mathcal{F}(v))| \leq (|A|-|A_v|)+|A_v|+(2|v|-4)(|A_v|-1) = 2(|v|-2)(|A_v|-1)+|A|$.

When $|v| < 3$, it can be checked directly that $|MF(\mathcal{F}(v))| \leq |A|+1$. $\bowtie$

Theorem 10 *Algorithm* MF-TRIE *runs in time $O(|v| \times |A|)$ on input word v if transition functions are implemented by transition matrices.*

References

1. A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search, *Comm. ACM* **18:6** (1975) 333–340.
2. M.-P. Béal, F. Mignosi, and A. Restivo. Minimal Forbidden Words and Symbolic Dynamics, in (C. Puech and R. Reischuk, eds., LNCS 1046, Springer, 1996) 555–566.
3. J. Cassaigne. Complexité et Facteurs Spéciaux, *Bull. Belg. Math. Soc.* **4** (1997) 67–88.
4. M. Crochemore, C. Hancart. Automata for matching patterns, in *Handbook of Formal Languages*, G. Rozenberg, A. Salomaa, eds.", Springer-Verlag", 1997, Volume 2, *Linear Modeling: Background and Application*, Chapter 9, 399–462.
5. V. Diekert, Y. Kobayashi. Some identities related to automata, determinants, and Möbius functions, Report 1997/05, Fakultät Informatik, Universität Stuttgart, 1997.
6. A. de Luca, F. Mignosi. Some Combinatorial Properties of Sturmian Words, *Theor. Comp. Sci.* **136** (1994) 361–385.
7. A. de Luca, L. Mione. On Bispecial Factors of the Thue-Morse Word, *Inf. Proc. Lett.* **49** (1994) 179–183.
8. R. McNaughton, S. Papert. *Counter-Free Automata*, M.I.T. Press, MA 1970.
9. D. Perrin. Symbolic Dynamics and Finite Automata, invited lecture in *Proc. MFCS'95*, LNCS **969**, Springer, Berlin 1995.