INFORMATIQUE THÉORIQUE ET APPLICATIONS

B. LE SAËC
I. LITOVSKY
B. PATROU

A more efficient notion of zigzag stability

Informatique théorique et applications, tome 30, n° 3 (1996), p. 181-194

<http://www.numdam.org/item?id=ITA_1996__30_3_181_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 30, n° 3, 1996, pp. 181-194)

A MORE EFFICIENT NOTION OF ZIGZAG STABILITY (*)

by B. Le Saëc ([1]), I. Litovsky ([2]) and B. Patrou ([1])

Communicated by J. Berstel

Abstract. – *In [6] a definition of zigzag stability for zigzag submonoids is given. We give here a new definition of zigzag stability, which is simpler and very close to the usual definition of stability for ordinary submonoids. We give a short proof that zigzag stability is decidable in the rational case. Moreover this notion of zigzag stability enables one to obtain an algorithm to decide whether a rational language is a zigzag code. Despite being in exponential time, the complexity of this algorithm is better than that of every other known algorithm [3], [8], [6].*

Résumé. – *Dans [6] une première définition de stabilité zigzag pour les sous-monoïdes zigzag est donnée. Nous donnons ici une autre définition de stabilité zigzag qui est plus simple et plus proche de la définition habituelle de stabilité pour les sous-monoïdes ordinaires. Par une preuve très courte, on montre que cette stabilité zigzag est décidable dans le cas rationnel. En outre elle permet d'obtenir un algorithme pour décider si un langage rationnel est un code zigzag ; algorithme dont la complexité est exponentielle mais meilleure que celle des autres algorithmes connus [3], [8], [6].*

1. INTRODUCTION

The zigzag operation $^{\uparrow}$ is an extension of the $*$ operation introduced in [1]. Let $X \subseteq A^*$ be a language. Every word $w \in X^*$ is obtained by a run on w with left-right steps in X, while a word $w' \in X^{\uparrow}$ can be obtained by a run on w' with left-right and also right-left steps in X. For instance if $X = \{abc, bc, bca\}$, the word $abca$ belongs to $X^{\uparrow}$ (but not to X^*); indeed it is obtained by the left-right step abc, the right-left step bc and the left-right step bca. That is the word $abca$ has a *zigzag factorization* (or z-factorization) on X.

(*) Received March 1994.

([1]) Laboratoire Bordelais de Recherche en Informatique, Unité de Recherche Associée au CNRS n° 1304, Université Bordeaux I, 351 cours de la Libération, 33405 Talence Cedex France. Email : (lesaec, patrou)@labri.u-bordeaux.fr

([2]) Laboratoire d'Informatique Signaux et Systèmes de Sophia Antipolis, Unité de Recherche Associée au CNRS n° 1376, Université de Nice Sophia Antipolis, route des colles, B.P. 145, 06903 Sophia Antipolis Cedex France. Email : lito@essi.fr

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/96/03/$ 4.00/

As for $*$ operation, one can define the following. A language C is a *z-code* if every word in A^* has at most one z-factorization on C [1]. A language M is a *z-submonoid* of A^* if $M = X^{\uparrow}$ for some language X, called *z-generator* of M [7]. Every z-submonoid M has a least z-generator called z-Root(M) and M is *z-free* if its z-Root is a z-code.

We are interested here in direct (*i.e.* without using the z-Root) characterizations for z-free z-submonoids. For free submonoids, stability is such a characterization [10]. A submonoid M is stable if:

$$\forall u, v, w \in A^* \ : \begin{matrix} u, vw \in M \\ uv, w \in M \end{matrix} \Big\} \Longrightarrow v \in M.$$

This stability property says that, in a free submonoid, every factorization (m_1, m_2) with two steps in M is a "partial view" of the factorization $(x_1, \ldots, x_n)$ in Root(M), that is : $x_1 \ldots x_i = m_1$ and $x_{i+1} \ldots x_n = m_2$ for some integer i, $1 \leq i \leq n$.

In the same way, a first definition of z-stability is given in [6] considering three-step z-factorizations in a z-submonoid. However this definition is rather tedious: three cases depending of different overlapping of two three-step z-factorizations are considered.

In this paper, using the notion of *strict z-prefix* we give a simpler definition of z-stability which is very close to the original stability definition. A word u is a *strict z-prefix* of uv if there exists a zigzag calculus of u with step in M on the support uv. For instance, with $X = \{abc, bc, bca\}$, word a is a strict z-prefix of $abcb$. Like stability definition, for any word $m \in M$ two-step factorizations (m_1, m_2) of word m are considered. However hypothesis "$m_1 \in M$" is replaced here by "m_1 is a strict z-prefix of m in M". Thus the following definition is obtained. A z-submonoid M is *z-stable2* if for all words $u, v, w \in A^*$:

$$\begin{matrix} u \text{ strict z-prefix of } uvw \text{ in } M,\ vw \in M \\ uv \text{ strict z-prefix of } uvw \text{ in } M,\ w \in M \end{matrix} \Big\} \Longrightarrow v \text{ strict z-prefix of } vw \text{ in } M.$$

Then z-stability2 and z-freeness properties are proved to be equivalent. Moreover we give a very short proof that z-stability2 is decidable in the rational case.

Next we are interested in the different time complexities (in the worst case) of three algorithms which decide whether a given rational language R is a z-code. Firstly we consider the Anselmo algorithm [3] which decides directly whether R is a z-code. On the other hand we consider algorithms

using the z-stability1 or the z-stability2: we decide whether the z-submonoid M generated by R is z-stable then we decide whether R is the z-root of M. We found that the complexities of the three algorithms are in exponential time (in the size m of the minimal automaton accepting R). However the complexity of the algorithm using the z-stability2, $O(2^{16m^2}) \times 2^{O(m)}$, is better than that using the z-stability1, $O(2^{40m^2}) \times 2^{O(m)}$, which is better than that of the direct algorithm in [3], $O((2^{4m}+m^2)! \times p^{(2^{4m}+m^2)})$, where p is the size of the alphabet. Moreover in [8] an algorithm for testing whether a **finite** set is a z-code is proved. The complexity of this algorithm is given in [8]: $O(2^{P(l)}n^{P(l)})$, where P is a polynomial of degree $2 \times maxl$ ($maxl$ is the maximal length of a word in R), l is the length of R and n is the cardinality of R. Thus this complexity is worst than the one of the previous algorithms using a z-stability property.

Definitions and notation are recalled in Section 2. Section 3 contains the proof that z-stability2 and z-freeness properties are equivalent. The rational case is studied in Section 4. Section 5 deals with the complexity issue.

2. PRELIMINARIES

Let A be an alphabet. As usual, A^* is the free monoid of all finite words over A, the empty word is denoted by ε and $A^+ = A^* \setminus \{\varepsilon\}$. The concatenation of two words $u, v \in A^*$ is denoted by uv. The notations $u \leq v$ or $u < v$ mean that u is a prefix of v (*i.e.* $v \in uA^*$) or a proper prefix of v (*i.e.* $v \in uA^+$) respectively.

For any language X, X^* denotes the submonoid of A^* generated by X. A *factorization* on X of a word w is a tuple $(x_1, ..., x_n)$ of words in X such that $x_1 ... x_n = w$. A language $X \subseteq A^*$ is a *code* if every word $w \in A^*$ has at most one factorization on X. If X is a code then X^* is a *free* submonoid of A^*.

For any alphabet A, we denote by $\overline{A}$ a disjoint alphabet in bijection with A. For every $a \in (A \cup \overline{A})$, we denote by $\overline{a}$ the element associated with a. For every word $x = a_1 \cdots a_n \in (A \cup \overline{A})^*$, $\overline{x}$ is the word $\overline{a}_n \cdots \overline{a}_1$. For every subset X of $(A \cup \overline{A})^*$, we denote $\overline{X} = \{\overline{x} \ : \ x \in X\}$ and we use the free monoid $(X \cup \overline{X})^*$ generated by $X \cup \overline{X}$. Then a word in the free monoid $(X \cup \overline{X})^*$ is denoted by a tuple $(x_1, \ldots, x_n)$ where every x_i is in $(X \cup \overline{X})$.

We denote by $\overset{X}{\longmapsto}$ the relation defined for all $u, v \in (X \cup \overline{X})^*$ by $u \overset{X}{\longmapsto} v$ if $u = f\alpha g$, $v = fg$ with $\alpha = (x, \overline{x})$ or $\alpha = (\overline{x}, x)$ for some $x \in X$. We call X-*reduction* the reflexo-transitive closure of $\overset{X}{\longmapsto}$. Then with every

word $w \in (X \cup \overline{X})^*$ is associated a unique X-reduced word $Red_X(w)$ ($Red_X(w)$ is the canonical representative of the class of w in the free group generated by X).

We denote by $\overset{l}{\longmapsto}$ the relation defined for all $u, v \in (X \cup \overline{X})^*$ by $u \overset{l}{\longmapsto} v$ if $u = f\alpha g$, $v = fg$ with $Red_A(\alpha) = \varepsilon$. We call *l-reduction* the reflexo-transitive closure of $\overset{l}{\longmapsto}$. Then with every word $w \in (X \cup \overline{X})^*$ is associated a set $Red_l(w)$ of l-reduced words. Note that the l-reduction is confluent iff X is a zigzag-code [5].

A *zigzag calculus* (z-calculus), $C_X(w_1, u, w_2)$, of a word $u \in A^*$ with context $(w_1, w_2) \in A^* \times A^*$ is a tuple $(x_1, ..., x_n) \in (X \cup \overline{X})^*$ such that:

1. $Red_A(x_1 ... x_n) = u$;
2. $\forall\ 1 \leq i \leq n,\ \varepsilon \leq Red_A(w_1 x_1 ... x_i) \leq w_1 u w_2$.

When furthermore:

3. $\forall\ 1 < i < n,\ \varepsilon < Red_A(w_1 x_1 ... x_i) < w_1 u w_2$

the z-calculus is called *strict*.

A *z-decomposition* on X of a word u is a strict z-calculus of u with context $(\varepsilon, \varepsilon)$. A *z-factorization* is a l-reduced z-decomposition; that is a z-decomposition $(x_1, ..., x_n)$ such that:

$$\forall\ 1 \leq i < j \leq n,\ Red_A(x_1 ... x_i) \neq Red_A(x_1 ... x_j).$$

In the sequel $f_{w,X}$ (or simply f_w) denotes a z-factorization of word w on X, it is drawn with full line, while a z-calculus is drawn with dashed line (*see* Figure 2 and Figure 1).

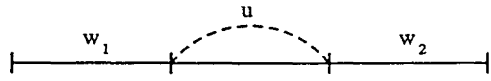

Figure 1. – A z-calculus of u with context (w_1, w_2).

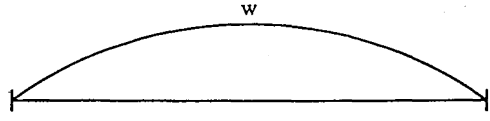

Figure 2. – A z-factorization of w.

Let $u, v \in A^*$, u is a *z-prefix* in X of uv if there exists a z-calculus $C(\varepsilon, u, v)$. If the z-calculus $C(\varepsilon, u, v)$ is strict, u is a *strict* z-prefix of uv. We denote by $Z\text{-}pref_X(w)$ (resp. $Z\text{-}pref\text{-}strict_X(w)$) the set of words $u \in A^*$ such that u is a z-prefix (resp. strict z-prefix) in X of word w.

The set $X^\uparrow$ of words having a z-factorization on X is called the *z-submonoid* of A^* generated by X [7]. Of course, $X^\uparrow$ is a submonoid of A^* which contains X^*. Let L be a z-submonoid of A^*, we call *z-Root* of L the set of words having exactly one z-factorization on L. A language X is a *z-code* if every word of $X^\uparrow$ has exactly one z-factorization on X [2]. A z-submonoid L is *z-free* if z-Root(L) is a z-code.

3. EQUIVALENCE BETWEEN Z-STABILITY AND Z-FREENESS

We give here a new definition of z-stability, which is simpler than the definition in [6].

DEFINITION 1: *Let M be a z-submonoid of A^*, M satisfies the property of* z-stability2 *if:*
$\forall u, v, w \in A^*$:

$$\left.\begin{array}{l} uv \in Z\text{-}pref\text{-}strict_M(uvw),\ w \in M \\ u \in Z\text{-}pref\text{-}strict_M(uvw),\ vw \in M \end{array}\right\} \Rightarrow v \in Z\text{-}pref\text{-}strict_M(vw)$$

(*see* Figure 3).

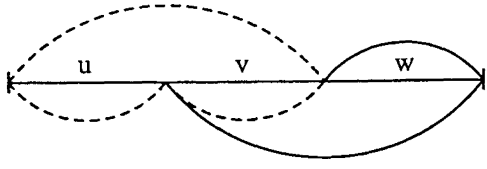

Figure 3. – The z-stabitity2.

Remark: The word *strict* cannot be removed in the previous definition, indeed v is always a z-prefix of vw since vw and w belong to M.

To prove that z-freeness and z-stability2 are equivalent, the two following results are used.

PROPOSITION 2 [5]: *Let $X \subseteq A^*$. Let $(x_1, \ldots, x_n)$ be a z-calculus of ε with context (w, w') such that $(x_1, \ldots, x_{n-1})$ is l-reduced and let $\overline{u}$ be the longest*

word in the set $\{Red_A(x_1 \dots x_i),\ 1 \leq i \leq n\} \cap \overline{A}^*$, *and* v *be the longest word in* $\{Red_A(x_1 \dots x_i),\ 1 \leq i \leq n\} \cap A^*$. *If* $n > 2$ *then the word* uv *has two distinct z-factorizations.*

COROLLARY 3 [7], [5]: *Let* $X \subseteq A^*$ *such that* X *is not a z-code. Let* w *be a word of minimal length having two distinct z-factorizations,* $(x_1, \dots, x_n)$ *and* $(y_1, \dots, y_m)$ *on* X. *Then* $x_1 \neq y_1$ *and* $x_n \neq y_m$.

We deduce the following lemma:

LEMMA 4: *A language* $X \subseteq A^*$ *is a z-code iff* $\forall u, v, w \in A^*$, v *has at most one l-reduced z-calculus* $C(u, v, w)$ *on* X.

Proof: If for all $u, v, w \in A^*$, v has at most one l-reduced z-calculus $C(u, v, w)$ in X, a fortiori every word $m \in X^{\uparrow}$ has at most one l-reduced z-calculus with context $(\varepsilon, \varepsilon)$. That is m has exactly one z-factorization on X. Conversely, if there exist $u, v, w \in A^*$ such that v has two distinct l-reduced z-calculus with context (u, w) : $C(u, v, w) = (x_1, \dots, x_p)$ and $C'(u, v, w) = (y_1, \dots, y_q)$. Without loss of generality, one can assume that $x_1 \neq y_1$. We consider $j = min\{1, \dots, p\}$ and $j' = min\{1, \dots, q\}$ such that $Red_A(x_1 \dots x_j) = Red_A(y_1 \dots y_{j'})$. Then the z-calculus of ε, $(x_1, \dots, x_j, \overline{y_{j'}}, \dots, \overline{y_1})$ is such that $(x_1, \dots, x_j, \overline{y_{j'}}, \dots, \overline{y_2})$ is l-reduced. As $j + j' > 2$ and $x_1 \neq y_1$, it follows from Proposition 2 that X is not a z-code. □

PROPOSITION 5: *Let* $M \subseteq A^*$ *be a z-submonoid. If* M *is z-free then* M *is z-stable2.*

Proof: Let $u, v, w \in A^*$ be three words satisfying the hypothesis of z-stability2. We denote by $C(\varepsilon, uv, w)$ the l-reduced strict z-calculus on z-Root(M) of uv with context (ε, w), f_w the z-factorization of w on z-Root(M), $C(\varepsilon, u, vw)$ the l-reduced strict z-calculus on z-Root(M) of u with context (ε, vw), f_{vw} the z-factorization of vw on z-Root(M). Then $C_1 = \overline{C}(\varepsilon, u, vw)C(\varepsilon, uv, w)$ is a z-calculus of v with context (u, w) such that word vw is never reached (since $C(\varepsilon, uv, w)$ and $C(\varepsilon, u, vw)$ are strict). As z-Root(M) is a z-code, C_1 may be l-reduced in a unique z-calculus, C_1', of v with context (u, w). On the other hand, $C_2 = f_{vw}\overline{f}_w$ is a z-calculus of v with context (ε, w). As z-Root(M) is a z-code, C_2 may be l-reduced in a unique z-calculus, C_2', of v with context (ε, w). Thus a fortiori, C_2' is a z-calculus of v with context (u, w), then by Lemma 4,

$C'_1 = C'_2$. That is, C'_1 is a z-calculus of v with context (ε, w) and word vw is never reached. Hence v is a strict z-prefix of vw. □

PROPOSITION 6: *Let $M \subseteq A^*$ be a z-submonoid. If M is z-stable2 then M is z-free.*

Proof: Assume that M is z-stable2 and is not z-free. Let $m \in M$ be a word of minimal length having two distinct z-factorizations on z-Root(M) (*see* Figure 4):

$f_m = (x_1, ..., x_p),\ \forall i \in \{1, \ldots, p\}\ x_i \in z\text{-Root}(M)$

$g_m = (y_1, ..., y_q),\ \forall j \in \{1, \ldots, q\}\ y_j \in z\text{-Root}(M)$

with $x_p \neq y_q$ (assume that x_p is a strict suffix of y_q).

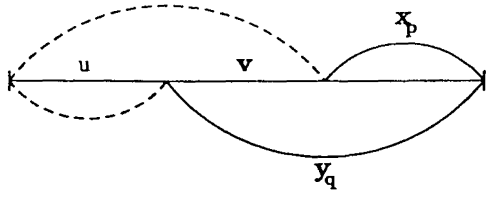

Figure 4.

We note $u = Red_A(y_1 ... y_{q-1})$, $v = y_q x_p^{-1}$ and $w = x_p$. For u, v, w the hypothesis of z-stability2 are satisfied, thus v is a strict z-prefix of vw. Hence $C(\varepsilon, v, w)w$ is a z-calculus of $vw = y_q$, a contradiction since $y_q \in z$-Root(M). □

Using a notion of *strict z-suffix*, we give a third definition of z-stability where "prefix" and "suffix" have a symetric part. Of course, one can prove that this definition is equivalent to the previous ones.

DEFINITION 7: Let M be a z-submonoid of A^*, M satisfies the property of *z-stability3* if:

$$\forall u, v, w \in A^*: \left.\begin{array}{l} uv \in Z\text{-}pref\text{-}strict_M(uvw), \\ w \in Z\text{-}suff\text{-}strict_M(uvw) \\ u \in Z\text{-}pref\text{-}strict_M(uvw), \\ vw \in Z\text{-}suff\text{-}strict_M(uvw) \end{array}\right\} \Rightarrow \text{there exists a strict}$$

z-calculus of v in M on the context (u, w)

(*see* Figure 5).

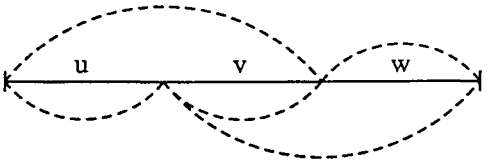

Figure 5.

4. RATIONAL CASE

In the rational case, z-stability2 property is of course decidable, since it is equivalent with z-freeness or z-stability property which are decidable properties [1], [6]. However the previous proofs are rather long. We give a short way to prove that z-stability2 property is directly decidable in the rational case.

Notation: for any marker $\#$ (*i.e.* $\#$ is a new letter $\notin A$):
$Z\text{-}pref_{\#}(M) = \{u\#v \ : \ u \in Z\text{-}pref_M(uv)\}$ and
$Z\text{-}pref\text{-}strict_{\#}(M) = \{u\#v : u \in Z\text{-}pref\text{-}strict_M(uv)\}$
$= Z\text{-}pref_{\#}(M)A^+.$

LEMMA 8: *If M is a rational language, then $Z\text{-}pref_{\#}(M)$ (and thus $Z\text{-}pref\text{-}strict_{\#}(M)$) are rational languages so.*

Proof: We give a rational expression for $Z\text{-}pref_{\#}(M)\$$ where $\#$ and $\$$ are two markers:
$Z\text{-}pref_{\#}(M)\$ = [M + (M \sqcup\!\sqcup \#) + \#A^*\$]^{\uparrow} \cap A^*\#A^*\$$ where $(M \sqcup\!\sqcup \#)$ is the set of words $\alpha\#\beta$ with $\alpha\beta \in M$.
Obviously, $Z\text{-}pref_{\#}(M)\$ \subseteq [M + (M \sqcup\!\sqcup \#) + \#A^*\$]^{\uparrow} \cap A^*\#A^*\$$. Conversely, if $w \in [M + (M \sqcup\!\sqcup \#) + \#A^*\$]^{\uparrow} \cap A^*\#A^*\$$, then every z-factorization, $f_w = (x_1, \ldots, x_n)$, of w on $M + (M \sqcup\!\sqcup \#) + \#A^*\$$ is such that: $\forall\ 1 \leq i \leq n-1 \ : \ x_i \in M + (M \sqcup\!\sqcup \#)$ and $x_n \in \#A^*\$$. Hence x_{n-1} or $\overline{x}_{n-1} \in M\#$, it follows that $w \in Z\text{-}pref_{\#}(M)\$$. Now as the operations $\sqcup\!\sqcup, {}^{\uparrow}, +, *, \cap, .$ preserve rational sets, we have the result. □

PROPOSITION 9: *One can decide whether a given rational z-submonoid is z-stable2.*

Proof: To decide whether a given rational z-submonoid M is z-stable2, it is sufficient to decide whether set M_1 is a subset of $Z\text{-}pref\text{-}strict_{\#}(M)$ where:

$M_1 = \{v\#w/\exists u \in A^* \text{ with: } u\#vw, uv\#w \in Z\text{-}pref\text{-}strict_{\#}(M) \text{ and } w, vw \in M\}$
(*see* Figure 6).

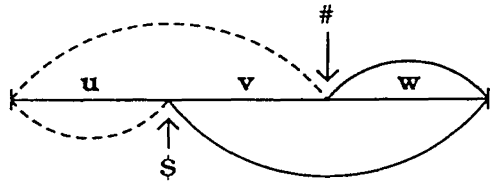

Figure 6.

We note:
$E_1 = (Z\text{-}pref\text{-}strict\#(M) \sqcup \$)$
$E_2 = (A^+\$A^+\#(M\backslash\varepsilon))$
$E_3 = (Z\text{-}pref\text{-}strict\$(M) \sqcup \#)$
$E_4 = (A^+\$(M \sqcup \#))$.

Then by construction: $M_1 = (A^+\$)^{-1}(E_1 \cap E_2 \cap E_3 \cap E_4)$. Hence the inclusion $M_1 \subseteq Z\text{-}pref\text{-}strict_{\#}(M)$ is decidable in the rational case. □

5. COMPLEXITY ISSUE

We now study the time complexity (in the worst case) of three algorithms to decide wether a given rational language $X \subseteq A^*$, is or not a z-code. The first algorithm was proposed by M. Anselmo in [3], the second one uses the previous algorithm to test the z-stability2 and the third one uses an other definition of z-stability previously given in [6].

To simplify we number the three algorithms: the algorithm given in [2] will be "algorithm 1", the algorithm defined in the previous sections will be "algorithm 2", and the algorithm described in [6] will be "algorithm 3".

Recall that we can construct a (flower) two-way automaton with $2n$ states recognizing the language $X^{\uparrow}$ from an automaton with n states recognizing a language X [3]. For algorithms 2 and 3 we use the following results concerning the blow-up in the number of automaton states to obtain (one-way) automata equivalent to two-way automata:

• given a (nondeterministic) two-way automaton with n states we can construct an equivalent nondeterministic automaton with $2^{O(n)}$ states ([9], [11]);

• given a (nondeterministic) two-way automaton with n states we can construct an equivalent deterministic automaton with $O(2^{n^2})$ states ([12]).

To conclude this section, we compare the complexities found for algorithms 1 to 3 and the complexity of the algorithm in [8] for testing whether a finite set is a z-code.

5.1. ALGORITHM 1

Method: Let $X \subseteq A^*$ be a rational language given by $\mathcal{A}_0$ a deterministic automaton recognizing X (we denote by m the state number of $\mathcal{A}_0$).

In [3] it is proved that $2^{4m} + m^2$ is an upper bound for the length of a shorter word having two different z-factorizations on X when X is not a z-code.

– We then determine the words number we will have to study.

– We look at all their potential z-factorizations.

We have to study the words in X^* with length $\leq 2^{4m} + m^2$. The number of these words is at most : $p + p^2 + p^3 + ... + p^{2^{4m}+m^2}$ (where p is the cardinality of the alphabet). To find the maximum number of potential z-factorizations of any word u with length n, we brutally proceed as this:

• there is at most 1 z-factorization of u in one step;

• there are at most $(n-1)$ z-factorizations of u in two steps: $(n-1)$ possibilities for the first step and the second one goes to the end of the word;

• there are at most $(n-1)(n-2)$ z-factorizations of u in three steps: $(n-1)$ possibilities for the first step, $(n-2)$ possibilities for the second step and the third one goes to the end of the word.

etc...

• there are at most $(n-1)(n-2)...2*1$ z-factorizations of u in n steps.

So we have $(n-1)! * \sum_{i=0}^{n-1} \left(\frac{1}{i!}\right)$ potential z-factorizations of u. For each one we have to determine whether it is really a z-factorization of u or not. So the number of operations we have to do is (in the worst case):

$$\sum_{n=1}^{2^{4m}+m^2} ([(n-1)! \sum_{i=0}^{n-1} (\frac{1}{i!})] \times p^n).$$

So the complexity of the first algorithm can be determine as this (by setting $f(n) = \sum_{k=1}^{n}(k-1)! \times p^k$):

$$\begin{aligned}\frac{f(n)}{(n-1)! \times p^n} &= \sum_{k=1}^{n-1}\left(\frac{(k-1)!}{(n-1)!} \times \frac{p^k}{p^n}\right) + 1 \\ &\leq \sum_{k=1}^{n-1}\left(\frac{(n-2)!}{(n-1)!} \times \frac{p^{n-1}}{p^n}\right) + 1 \\ &\leq \frac{1}{p} + 1 \\ &\leq 2.\end{aligned}$$

So we have: $(n-1)! \times p^n \leq f(n) \leq 2 * (n-1)! \times p^n$, that is: $f(n) \in O((n-1)! \times p^n)$.

Thus we have the following result:

PROPOSITION 10: *Using the test for z-code in [3], one can decide whether a rational language, given by a deterministic automaton with m states, is a z-code in $O((2^{4m} + m^2)! \times p^{(2^{4m}+m^2)})$ time (where p is the alphabet size).*

5.2. ALGORITHM 2

Method: Let $X \subseteq A^*$ be a rational language given by $\mathcal{A}_0$ a deterministic automaton recognizing X (we denote by m the state number of $\mathcal{A}_0$).

- We verify that $X^{\uparrow}$ is z-stable2.
- We verify that $X = z\text{-Root}(X^{\uparrow})$.

To verify that $X^{\uparrow}$ is z-stable2 we have to test an inclusion $M_1 \subseteq Z\text{-}pref\text{-}strict_{\#}(M)$ (with the notation of proof of Proposition 9). Thus it is sufficient to have a deterministic automaton recognizing Z-pref-strict$_{\#}(X^{\uparrow})$ and a non-deterministic automaton recognizing M_1.

To compute an automaton recognizing Z-pref$_{\#}(X^{\uparrow}).\$ = (X + (X \sqcup \#) + \#A^*\$)^{\uparrow} \cap A^*\#A^*\$$, the sizes of the different automata used are the following:

- for $(X + (X \sqcup \#) + \#A^*\$)$: $2(m+1)$;
- for $(X + (X \sqcup \#) + \#A^*\$)^{\uparrow}$: $O(2^{16(m+1)^2})$ (deterministic) , or $2^{O(m)}$ (nondeterministic);
- for $A^*\#A^*\$$: 3 states.

Thus Z-pref$_{\#}(X^{\uparrow})$ may be recognized by an automaton having a state number in:

- $O(2^{(16m^2+32m)})$ for the deterministic case;
- $2^{O(m)}$ for the nondeterministic case.

By definition of Z-pref-strict$_{\#}(X^{\uparrow})$, automata with the same respective sizes may recognize Z-pref-strict$_{\#}(X^{\uparrow})$.

Building the four sets E_1, E_2, E_3, E_4 may be as follows:

$E_1 = Z$-pref-strict$_{\#}(X^{\uparrow}) \sqcup\!\sqcup \$: 2^{O(m)}$ (we do not need determinism here).

$E_2 = A^+\$A^+\#(X^{\uparrow}\backslash\varepsilon) : 2^{O(m)}$.

E_3 (idem E_1).

$E_4 = A^*\$(X^{\uparrow} \sqcup\!\sqcup \#)$: $2^{O(m)}$.

Thus $E = E_1 \cap E_2 \cap E_3 \cap E_4$ is computed with an automaton having $2^{O(m)}$ states, and also for $M_1 = (A^+\$)^{-1}E$.

Hence $M_1 \subseteq Z$-pref-strict$_{\#}(X^*)$ can be verified by an automaton having $O(2^{16m^2}) \times 2^{O(m)}$ states.

For the second point of the method, we consider the set 2-fact$(X^{\uparrow})$ of words in $X^{\uparrow}$ having one non-trivial z-factorization on $X^{\uparrow}$. So we have z-Root$(X^{\uparrow}) = (X^{\uparrow}\backslash$ 2-fact$(X^{\uparrow}))$. On the other hand, the set 2-fact$(X^{\uparrow})$ is represented by the following expression:

2-fact$(X^{\uparrow}) = (X^{\uparrow} \sqcup\!\sqcup \#)\$A^+ \cap A^+\#(X^{\uparrow} \sqcup\!\sqcup \$) \cap A^+\#X^{\uparrow}\$A^+$ (*see* Figure 7).

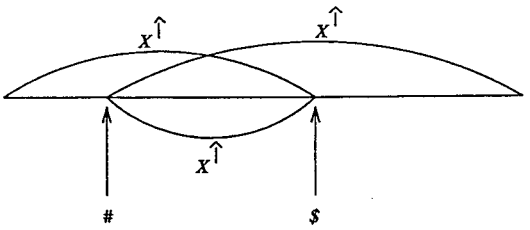

Figure 7.

We now determine the states number of the minimal automaton recognizing 2-fact$(X^{\uparrow})$:

- for $(X^{\uparrow} \sqcup\!\sqcup \#)\$A^+$: $O(2^{4m^2})$;
- for $A^+\#(X^{\uparrow} \sqcup\!\sqcup \$)$: $O(2^{4m^2})$;
- for $A^+\#X^{\uparrow}\$A^+$: $O(2^{4m^2})$.

Thus for 2-fact $(X^{\uparrow})$: $O(2^{12m^2})$.

As z-Root $(X^{\uparrow}) \subseteq X$, we have only to test whether $X \cap 2\text{-fact}(X^{\uparrow}) = \emptyset$, this is made in $O(m \times 2^{12m^2})$ time.

Thus we have the following result:

PROPOSITION 11: *Using the z-stability2, one can decide whether a rational language, given by a deterministic automaton with m states, is a z-code in $O(2^{16m^2}) \times 2^{O(m)}$ time.*

5.3. ALGORITHM 3

Method: Let $X \subseteq A^*$ be a rational language given by $\mathcal{A}_0$ a deterministic automaton recognizing X (we denote by m the state number of $\mathcal{A}_0$).

– We verify that $X^{\uparrow}$ is z-stable in the sense of by studying the three properties of z-stability [6].

– We verify that $X = z\text{-Root}(X^{\uparrow})$ (as in Algorithm 2).

We have to test three conditions for $X^{\uparrow}$ to decide whether a given rational language X is z-stable [6]. A deterministic automaton $\mathcal{A}_1$ recognizing $X^{\uparrow}$ has $O(2^{(4m^2)})$ states. To decide the first two conditions, we make a product $\mathcal{A}_2$ of six automata $\mathcal{A}_1$ and we have to test whether a language recognized by $\mathcal{A}_2$ is a subset of $X^{\uparrow}$: this is made in $O(2^{(28m^2)})$ time.

To decide the third condition, we make yet a product $\mathcal{A}_3$ of six automata $\mathcal{A}_1$, and we have to test whether a language recognized by $\mathcal{A}_3$ is a subset of $Z\text{-pref}_{\#}(X^{\uparrow})$: this is made in $O(2^{(40m^2+8m)})$ time.

The verification that $X = \text{z-Root}(X^*)$ is made as previously.

PROPOSITION 12: *Using the z-stability defined in [6], one can decide whether a rational language, given by a deterministic automaton with m states, is a z-code in $O(2^{40m^2}) \times 2^{O(m)}$ time.*

5.4. CONCLUSION

We can easily note that the three complexities found are "strongly" exponential. However according to the results, the z-stability2 leads to the less inefficient algorithm and the direct method from [3] gives the more inefficient algorithm (maybe for this algorithm, the given bounds are not optimal). Moreover using the z-stability2 the complexity seems to be better than the one found in [8] for the paticular case of finite languages X :

$O(2^{P(l)}n^{P(l)})$, where P is a polynomial of degree $2 \times maxl$ ($maxl$ is the maximal length of a word in X), l is the length of X and n is the cardinality of X.

REFERENCES

1. M. ANSELMO, Automates et codes zigzag, *R.A.I.R.O. Theoretical Informatics and Applications*, 1991, *25*, 1, pp. 49-66.
2. M. ANSELMO, Sur les codes zigzag et leur décidabilité, *Theoretical Computer Science*, 1990, *74*, pp. 341-354.
3. M. ANSELMO, *Automates bilatères et codes zigzag*, Thèse L.I.T.P. 90-27, 1990.
4. J. BERSTEL and D. PERRIN, *Theory of codes*, Academic Press, 1985.
5. DO LONG VAN, B. LE SAËC and I. LITOVSKY, On coding morphisms for zigzag codes, *R.A.I.R.O. Theoretical Informatics and Applications*, 1992, *26*, 6, pp. 565-580.
6. DO LONG VAN, B. LE SAËC and I. LITOVSKY, Stability for the zigzag submonoids, *Theoretical Computer Science*, 1993, *108*, pp. 237-249.
7. M. MADONIA, S. SALEMI and T. SPORTELLI, On z-submonoids and z-codes, *R.A.I.R.O. Theoretical Informatics and Applications*, 1991, *25*, 4, pp. 305-322.
8. M. MADONIA, S. SALEMI and T. SPORTELLI, A generalization of Sardinas-Patterson algorithm to z-codes, *Theoretical Computer Science*, 1993, *108*, pp. 251-270.
9. M. O. RABIN and D. SCOTT, Finite automata and their decision problems, *IBM J. Res. Develop.*, 1959, *3*, pp. 114-125.
10. M. P. SCHUTZENBERGER, *Une théorie algébrique du codage*, Séminaire Dubreil-Pisot, 1955-1956, Exposé No. 15.
11. J. C. SHEPHERDSON, The reduction of two-way automata to one-way automata, *IBM J. Res. Develop.*, 1959, *3*, pp. 199-201.
12. M. Y. VARDI, A note on the reduction of two-way automata to one-way automata, *Information Processing Letters*, 1989, *30*, pp. 261-264.