

Reduced languages as ω -generators

Sandrine JULIA¹ and TRAN Vinh Duc²

¹ Université de Nice - Sophia Antipolis,
Laboratoire I3S - CNRS, B.P. 121,
06903 Sophia Antipolis Cedex, France.
Sandrine.Julia@unice.fr

² Thang Long School of Technology,
Hanoi, Viet-Nam.
tvduc@ifi.edu.vn

Abstract. We consider the following decision problem: “Is a rational ω -language generated by a code ?” Since 1994, the codes admit a characterization in terms of infinite words. We derive from this result the definition of a new class of languages, the reduced languages. A code is a reduced language but the converse does not hold. The idea is to “reduce” easy-to-obtain minimal ω -generators in order to obtain codes as ω -generators.

Introduction

Our research deals with the classical theory of automata and languages. We particularly focus on the rational languages of infinite words (ω -languages) which are recognized by Büchi or Muller automata [16]. A rational ω -language may be ω -generated by a language. The operation $^\omega$ stands for the infinite concatenation and maps a language L into an ω -language L^ω . L^ω is then called an ω -power and L is one of its ω -generators. One can decide if a rational ω -language admits an ω -generator. If so, a rational ω -generator exists [14]. Various decision problems arise from the set of ω -generators of a given rational ω -language.

Here is the open decision problem on which we focus: “Is a rational ω -language generated by a code ?” A language L is a code if and only if every non-empty word in L^* has a unique factorization over L [2]. The similar problem for the Kleene closure $*$ instead of $^\omega$ has a simple solution. The monoid L^* is its own greatest generator and is generated by a code if and only if its root $L^* \setminus (L^* \setminus \{\varepsilon\})^2$ is a code. By analogy, we wonder when a rational ω -language L^ω is the ω -power of a code.

Unfortunately, the set of ω -generators of a rational ω -language does not admit one but a finite number of maximal ω -generators [14]. Even if the greatest ω -generator exists, the ω -power can be generated by a code without the root of its greatest ω -generator being a code. For instance, consider the ω -power L^ω with the root of its greatest ω -generator equals to $L = a + ab + ba$, which is not a code. Surprisingly, L^ω is ω -generated by the infinite code $C = a + (ab)^*ba$.

Our approach of the problem consists of the definition of a new class of ω -languages called the *reduced languages*. It is known that a code is minimal

(with respect to inclusion) in the set of ω -generators. Our new class of reduced languages is useful here because it contains the codes and is included in the set of minimal ω -generators. Usual approaches restrict the problem to subclasses of codes: prefix codes [13], ω -codes [9], codes with delay [5]. In addition, the problem is solved for prefix codes in [13]. Here, we decide to widen the problem by considering a notable superclass: the class of *reduced languages*.

The graph of a Büchi automaton reveals that it is possible to slightly modify an ω -generator without changing the ω -power expressed. But in practice, we had no idea about how to modify a minimal ω -generator towards another which could be a code. Hopefully, to get a reduced language is possible, and may provide codes.

The paper is divided in five main sections. The two first ones set preliminary definitions and useful results, in particular, the characterization of codes by means of infinite words [6]. The third introduces the concept of reduced languages, followed by the study of the whole class and its decidability. Different cases are then detailed in the fourth section to convince that, despite of their great similitary, the reduced languages do not behave exactly as codes when taken as ω -generators. At last, the fifth section explores the ability of the construction of reduced ω -generators to reach codes.

1 Preliminaries

Let Σ be a finite alphabet. A *word* (resp. ω -*word*) is a finite (resp. infinite) concatenation of letters in Σ . We note ε the empty word. Σ^* is the set of words over Σ , $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. Σ^ω is the set of ω -words. Any subset of Σ^* is called a *language* and any subset of Σ^ω is called an ω -*language*.

A word u is a *prefix* of v if $v \in u(\Sigma^* \cup \Sigma^\omega)$ and we write: $u < v$. The induced order is the *prefix order*. For $v \in (\Sigma^* \cup \Sigma^\omega)$, $Pref(v)$ stands for the set of all prefixes of v . Hence, for every $L \subseteq (\Sigma^* \cup \Sigma^\omega)$, $Pref(L)$ is the set of the prefixes of the words in L .

Let $L \subseteq \Sigma^*$ be a language, the language L^* is the set of words built with words in L : $L^* = \{\varepsilon\} \cup \{a_1 \dots a_n \mid \forall i \ 1 \leq i \leq n, a_i \in L\}$. In the same way, the ω -*power* L^ω is the set of ω -words: $L^\omega = \{a_1 \dots a_n \dots \mid \forall i > 0, a_i \in L \setminus \{\varepsilon\}\}$. L^* (resp. L^ω) is generated (resp. ω -generated) by L , and so L is called a *generator* (resp. ω -*generator*). Henceforth, *minimality* or *maximality* are specifically used with respect to inclusion over the set of ω -generators.

Both of following languages are useful: $Prem(L) = (L \setminus \{\varepsilon\}) \setminus (L \setminus \{\varepsilon\})(L \setminus \{\varepsilon\})^+$ and, whenever M is a monoid, $Root(M) = Prem(M) = (M \setminus \{\varepsilon\}) \setminus (M \setminus \{\varepsilon\})^2$.

The *stabilizer* of L^ω is the language: $Stab(L^\omega) = \{u \in \Sigma^+ \mid uL^\omega \subseteq L^\omega\}$. $Stab(L^\omega)$ is a semigroup in which every ω -generator of L^ω is included. So, when $Stab(L^\omega)$ is an ω -generator of L^ω , it is the greatest [14]. The *characteristic language* of L^ω is the language: $\chi(L^\omega) = \{u \in \Sigma^+ \mid uL^\omega \subseteq L^\omega \text{ and } u^\omega \in L^\omega\}$. $\chi(L^\omega)$ is not anymore a semigroup. Every ω -generator of L^ω is still included in $\chi(L^\omega)$ and so, when $\chi(L^\omega)$ is an ω -generator of L^ω , it is also the greatest [14]. L^ω rational implies that $Stab(L^\omega)$ and $\chi(L^\omega)$ are also rational.

Let $L = L \setminus \{\varepsilon\}$ (abusively written $L = L \setminus \varepsilon$). A L -factorization of a word u in L^+ is a finite sequence of words in L : (u_1, u_2, \dots, u_n) such that $u = u_1 u_2 \dots u_n$. A L -factorization of an ω -word w in L^ω is an infinite sequence: $(w_1, w_2, \dots, w_n, \dots)$ such that $w = w_1 w_2 \dots w_n \dots$. We will say indifferently L -factorization or factorization over L . A language L is a *code* (resp. an ω -*code*) if every word $u \in \Sigma^*$ (resp. every ω -word $w \in \Sigma^\omega$) has at most one L -factorization [2][17]. Any ω -code is *a fortiori* a code. Later, rational languages and ω -languages can be denoted by their regular (ω)-expressions.

Let L be a language, the *adherence* of L is the ω -language $Adh(L) = \{w \in \Sigma^\omega \mid Pref(w) \subseteq Pref(L)\}$ and an ω -language A is an adherence if $A = Adh(L)$ for some language L .

A language L is said to have a *bounded (deciphering) delay* if: $\exists d \geq 0 \forall u \in L (uL^d \Sigma^\omega \cap L^\omega) \subseteq uL^\omega$.

A *finite automaton* \mathcal{A} (resp. *Büchi automaton* \mathcal{B}) is specified by $(\Sigma, Q, \delta, I, T)$ where Σ denotes the finite alphabet, Q the finite set of states, $I \subseteq Q$ the set of initial states and $T \subseteq Q$ the set of recognition states. A *run* of a word m in \mathcal{A} (resp. a *run* of an ω -word w in \mathcal{B}) is a finite sequence $l = (q_i)_{0 \leq i \leq n}$ (resp. an infinite sequence $l = (q_i)_{i \geq 0}$) of states in Q such that $q_0 \in I$ and $\forall i \delta(q_i, m_{i+1}) = q_{i+1}$, with m_i the i^{th} letter of m (resp. w). A word m (resp. an ω -word w) belongs to the language recognized by \mathcal{A} (resp. ω -language recognized by \mathcal{B}) if there exists a run $(q_i)_{0 \leq i \leq n}$ (resp. a run $(q_i)_{i \geq 0}$) such that $q_0 \in I$ and $q_n \in T$ (resp. $q_0 \in I$ and $Inf(w) \cap T \neq \emptyset$, where $Inf(w) = \{q \in Q / Card(\{i / q_i = q\}) \text{ is infinite}\}$). We note $L(\mathcal{A})$ (resp. $L(\mathcal{B})$) the language (resp. ω -language) recognized by \mathcal{A} (resp. \mathcal{B}). The set of recognized languages coincide with the set of rational languages. A rational ω -language is of the form: $L = \bigcup_{i=1}^n A_i B_i^\omega$ with $n \geq 1$ such that for every i , A_i and B_i are rational languages respectively included in Σ^* and Σ^+ . Their class coincides with the class of ω -languages recognized by Büchi automata [18].

2 Useful results

In this section, we present some preliminary results. The first one is very important for our purpose. It gives an elegant characterization of codes based on periodic infinite words.

Proposition 1. [6] *Let $L \subseteq \Sigma^+$. The language L is a code if and only if for every word $u \in L^+$, u^ω has a unique L -factorization.*

Below, we recall two results about adherence needed in the sequel to justify the hypothesis taken.

Proposition 2. [3][11] *Let L be a language. If L^ω is an adherence, then*

$$L^\omega = Adh(L^*) = Adh(Pref(L^*)).$$

Proposition 3. [14] *Let L be a language. If L^ω is an adherence then $\chi(L^\omega)$ and $Stab(L^\omega)$ coincide with the greatest ω -generator of L^ω .*

Example 1. Consider $L = a + ab + b^2$. L^ω is finitely ω -generated so it is an adherence and $\chi(L^\omega) = \text{Stab}(L^\omega) = L^+$ is the greatest ω -generator. Let $K = a^*b$. K^ω is not an adherence, $\text{Stab}(L^\omega) = \Sigma^+$ is not ω -generator of K^ω but $\chi(L^\omega) = \Sigma^*b\Sigma^*$ is the greatest ω -generator. Finally, let $M = \Sigma^*(aa + bb)$. There are two maximal ω -generators $M_1 = \Sigma^*(aa + bb)\Sigma^* + a(ba)^*$ and $M_2 = \Sigma^*(aa + bb)\Sigma^* + b(ab)^*$. Neither $\text{Stab}(L^\omega) = \Sigma^+$ nor $\chi(L^\omega) = M_1 \cup M_2$ are ω -generators of M^ω .

The following result is about languages with a bounded delay. Usually, this deciphering property is linked to codes, not here.

Proposition 4. [7] *Let L be a language with a bounded delay such that L^+ is the greatest ω -generator of L^ω . If L^ω is an adherence, then every ω -generator code of L^ω is necessarily a finite ω -code.*

We point out here the result called *lemma of infinite iteration* frequently used to prove the equality between two ω -powers.

Lemma 1. [14] *Let L and $R \subseteq \Sigma^+$ be two rational languages, $L^\omega \subseteq RL^\omega \Rightarrow L^\omega \subseteq R^\omega$.*

The language $L \setminus L\text{Stab}(L^\omega)$ is still an ω -generator of L^ω and will be useful to finally simplify ω -generators.

Proposition 5. [12] *Let L be a language, the following properties hold:*

- (i) $L \setminus L\text{Stab}(L^\omega) \subseteq \text{Prem}(L)$.
- (ii) $L \setminus L\text{Stab}(L^\omega)$ and $\text{Prem}(L)$ are ω -generators of L^ω .

At last, let us recall now a classic result on words.

Lemma 2. [15] *Two words $u, v \in \Sigma^+$ commute, i.e. $uv = vu$, if and only if there exists a word $z \in \Sigma^+$ and two different integers i and $j \geq 1$ such that $u = z^i$ and $v = z^j$.*

3 Reduced languages

In this section, we present a new class of languages based on a property particularly relevant when referring to ω -generators. This class lies between the class of codes and the class of minimal ω -generators. We call it the class of *reduced languages*.

3.1 Presentation

In the sequel, we present the definition of reduced languages which involves periodic ω -words. Then, we state a characterization of them in order to locate reduced ω -generators among minimal ω -generators.

Definition 1. A language $R \subseteq \Sigma^+$ is called *reduced* if:

$$\forall u \in R \quad u^\omega \notin (R \setminus u)^\omega$$

Proposition 6. Every reduced ω -generator is a minimal ω -generator.

Proof. Let L be a language. If L is not minimal, then there exists a word $u \in L$ such that $u^\omega \in L^\omega = (L \setminus u)^\omega$. Hence L is not reduced. \square

The converse does not hold. For instance, the language $L = a + ab + ba$ is minimal but is not reduced. Clearly, $(ab)^\omega \in (L \setminus ab)^\omega$.

Proposition 7. A language L is reduced if and only if for each word $u \in L$, the periodic ω -word u^ω has a unique L -factorization.

Proof. The second condition clearly implies L reduced. Conversely, assume there exists $u \in L$ such that u^ω has two L -factorizations with different first steps: (u, u, \dots) and (v_0, v_1, \dots) . Two cases arise:

- either, for each integer $i \geq 0$, $v_i \neq u$, hence $u^\omega \in (L \setminus u)^\omega$.
- either there exists a smallest integer $k > 0$ verifying $v_k = u$.

Hence, there exist two words $\alpha, \beta \in \Sigma^*$ and $n > 0$ such that:

- $v_0 \dots v_{k-1} \alpha = u^n$
- $v_0 \dots v_{k-1} u = u^n \beta$
- $v_0 \dots v_{k-1} u \alpha = u^n \beta \alpha = u^{n+1}$

then, $u = \alpha\beta = \beta\alpha$. According to Lemma 2, there exists z verifying $\alpha = z^i$ and $\beta = z^j$. We obtain $u^\omega = z^\omega = (v_0 \dots v_{k-1})^\omega$ and so, $u^\omega \in (L \setminus u)^\omega$.

In both cases, a contradiction appears with L reduced. \square

Table 1 gives the maximal number of factorizations of different kinds of ω -words, like in [6]. The asterisk $*$ attests that the column property characterizes the corresponding class of languages. Consequently:

Proposition 8. A code is a reduced language.

The converse does not hold. For instance, the language $L = a + ab + bc + c$ is a reduced language but is not a code since the ω -word $(abc)^\omega$ has two distinct L -factorizations. We summarize below the relations between the different classes of ω -generators we consider:

$$\text{Code } \omega\text{-generator} \Rightarrow \text{reduced } \omega\text{-generator} \Rightarrow \text{minimal } \omega\text{-generator.}$$

Table 1. Maximal number of factorizations over ω -codes, codes, reduced languages.

Language L	$\frac{u^\omega}{(u \in L)}$	$\frac{u^\omega}{(u \in L^+)}$	any
ω -code	1	1	1*
code	1	1*	∞
reduced language	1*	∞	∞

3.2 Decidability

The aim of this part is to ensure that the property of reduced language is decidable over the set of rational languages. Four preliminary lemmas are needed before stating the main result.

Let $L \subseteq \Sigma^+$ a language. We use the set $Amb(L)$ introduced in [10]. We restrict $Amb(L)$ to ω -words, so the set $Amb(L)$ contains ω -words in L^ω with several L -factorizations with different first steps.

$$Amb(L) = \{w \in L^\omega \mid \exists (w_i)_{i \in \mathbb{N}}, (w_j')_{j \in \mathbb{N}} \text{ two } L\text{-factorizations of } w \text{ with } w_0 \neq w_0'\}$$

Lemma 3. [6][9] *If a language $L \subseteq \Sigma^+$ is rational, then the set $Amb(L)$ is rational too.*

Proof. If L is rational, the congruence defined as $u \simeq v \Leftrightarrow u^{-1}L = v^{-1}L$ has a finite index. Let us write $\langle u \rangle$ the equivalence class of the word u . The set $Amb(L)$ is obtained as: $Amb(L) = \bigcup_{\langle u \rangle \subseteq L} \langle u \rangle (L^\omega \cap (u^{-1}L \setminus \varepsilon)L^\omega)$. \square

The following lemma is a consequence of Definition 1:

Lemma 4. *Let $L \subseteq \Sigma^+$ be a language, L is reduced if and only if L verifies:*

$$\forall u \in L \quad u^\omega \notin Amb(L)$$

We present here some notation concerning the Büchi congruence [4] in order to prove our result. Let $\mathcal{A} = (\Sigma, Q, I, \delta, T)$ a complete Büchi automaton. For each state $q \in Q$, and for every word $u \in \Sigma^*$, we write:

$$\delta_T(q, u) = \{q' \in Q \mid \text{exists } t \in T \text{ and } u_1, u_2 \in \Sigma^* \text{ with } u = u_1u_2 \text{ and } t \in \delta(q, u_1) \text{ and } q' \in \delta(t, u_2)\}$$

The Büchi congruence \approx is defined by:

$$u \approx v \Leftrightarrow \forall q \in Q, \begin{cases} \delta(q, u) = \delta(q, v) \\ \delta_T(q, u) = \delta_T(q, v) \end{cases}$$

for every $u, v \in \Sigma^+$. Let $[u] = \{w \in \Sigma^+ \mid w \approx u\}$ be the equivalence class of u . As \approx has a calculable finite index, we obtain:

Lemma 5. [4] *For each $u \in \Sigma^+$, its equivalence class $[u]$ is a constructible rational language.*

Lemma 6. *If $v_1 \approx v_2$, then $v_1^\omega \in L_\omega(\mathcal{A}) \Leftrightarrow v_2^\omega \in L_\omega(\mathcal{A})$.*

It is time to state the main result.

Theorem 1. *One can decide whether a rational language is a reduced language.*

Proof. Let L be a rational language. It is effective to:

- construct the automaton \mathcal{A} which recognizes the set $Amb(L)$ (according to Lemma 3);
- compute the equivalence classes $[u_1], \dots, [u_k]$ (according to Lemma 5);
- verify if there exists $[u_i]$ such that $[u_i] \cap L \neq \emptyset$ and $u_i^\omega \in Amb(L)$.

If so, L is not reduced, otherwise, L is reduced (according to lemmas 4 and 6).

\square

4 Reduced languages as ω -generators

The class of reduced languages comes from considerations on the set of ω -generators. This set contains or not a reduced language. If so, this set contains or not a code. Both subsections illustrate the main two cases, the first revealing incidently that a rational ω -power is not necessarily ω -generated by a code.

4.1 No reduced ω -generator

We show that there exists an ω -power that cannot be generated by a reduced language. Consequently, this implies that a rational ω -power is not necessarily generated by a code. Previously, this result has been proved in [19] and clearly reinforces the interest in the decision problem we study.

Proposition 9. *Some rational ω -powers do not admit reduced ω -generators.*

Proof. Consider $L = a^2 + a^3 + ba + b$. Notice that $L^\omega = \Sigma^\omega \setminus ab\Sigma^\omega$ and that $L = \chi(L^\omega)$ is the greatest ω -generator of L^ω . Assume that there exists a reduced ω -generator R of L^ω . Let us prove the following two facts:

Fact 1 *Let $w \in L^\omega$. If $w \in a\Sigma^\omega$ then $aw \in L^\omega$.*

Proof (Fact 1). Clearly, $w \in (a^2L^\omega \cup a^3L^\omega)$. If $w \in a^2L^\omega$ then $aw \in a^3L^\omega \subseteq L^\omega$. If $w \in a^3L^\omega$ then $aw \in (a^2)^2L^\omega \subseteq L^\omega$. \square

Fact 2 *For all $k \geq 1$ and $u \in \Sigma^*$, we obtain $\{a^k u, a^k u a\} \not\subseteq R$.*

Proof (Fact 2). If $\{a^k u, a^k u a\}$ is included in R , using Fact 1, we get:

$$(a^k u a)^\omega = (a^k u) \underbrace{(a^{k+1} u)^\omega}_{\in R^\omega}$$

Consequently, $(a^k u a)^\omega$ has two R -factorizations: $(\alpha_i)_{i \geq 0}$ and $(\beta_j)_{j \geq 0}$ with $\alpha_0 = a^k u a$ and $\beta_0 = a^k u$. Hence, R is not reduced. \square

- as $a^\omega \in L^\omega$, there exists a unique $i_0 > 1$ such that $a^{i_0} \in R$ (according to Fact 2 with $u = \varepsilon$).
- as $a^{i_0} a b a^\omega \in L^\omega$ and $a b a^\omega \notin L^\omega$, there exists a unique integer $i_1 \geq 0$ such that $a^{i_0} a b a^{i_1} \in R$.
- as $a^{i_0} a b a^{i_1} a b a^\omega \in L^\omega$ and $a^{i_0} a b a^{i_1} a \notin R$ (according to Fact 2), then, there exists a unique integer $i_2 \geq 0$ such that $a^{i_0} a b a^{i_1} a b a^{i_2} \in R$.
- and so forth, we define a unique infinite sequence $(i_j)_{j \geq 0}$.

Now, let us consider the following ω -word: $w = a^{i_0} a b a^{i_1} a b \dots a b a^{i_n} \dots$. This word w belongs to L^ω but lacks a factorization over R . We deduce that R is not an ω -generator of L^ω . We conclude that L^ω does not have any reduced ω -generator. \square

Corollary 1. *Some rational ω -powers do not admit codes as ω -generators.*

4.2 Reduced vs code ω -generator

In this section, we show that an ω -power generated by a reduced language is not necessarily generated by a code.

Proposition 10. *A rational language ω -generated by a reduced language is not necessarily ω -generated by a code.*

Proof. $L = a+ab+bc+c$ is a language with a bounded delay 1, studied in [1]. L^ω is an adherence since L is finite and then $\chi(L^\omega) = L^+$ is its greatest ω -generator. It is clear that L is reduced. Let us show that L^ω cannot be generated by a code. Assume firstly that C is an ω -code ω -generator of L^ω . As $(ua^i)a^\omega = (u)\underbrace{a^\omega}_{\in L^\omega}$ and

$(ub)c^\omega = (u)\underbrace{bc^\omega}_{\in L^\omega}$, we obtain the property P : $\{ua^i, u\} \not\subseteq C$ and $\{ub, u\} \not\subseteq C$, for

every $u \in \Sigma^+$ and $i > 0$. We intend to construct an infinite sequence of elements from C :

- as $a^\omega \in L^\omega$, there exists a unique integer $i_0 > 0$ (according to P) such that $a^{i_0} \in C$;
- $i_0 > 0$, then $a^{i_0}ba^\omega \in L^\omega$, but $a^{i_0}b \notin C$ (according to P), and there exists a unique integer $i_1 > 0$ such that $a^{i_0}ba^{i_1} \in C$;
- and so on; we define a unique infinite sequence $(i_j)_{j \geq 0}$.

The cardinality of C is necessarily infinite, so there is no finite ω -code ω -generating L^ω . According to Prop. 4, there is no code C ω -generating L^ω . \square

5 Reducing ω -generator

For the moment, the interest of the new class of reduced languages is not proven. However, some minimal ω -generators which are not codes are prevented from being codes *essentially* because there are not reduced. So, we present a method in order to make ω -generators reduced without affecting their ω -power.

5.1 Reduction

The reduction mixes two ideas: the first is a transformation required to aim at the uniqueness of factorizations of specific periodic ω -words, according to the characterization of reduced languages (Prop. 7). The second one is a simplification to guarantee the minimality of reduced ω -generators (Prop. 6).

Let us call $A(L)$ the language of words in L which prevents L from being reduced.

$$A(L) = \{u \in L \mid u^\omega \in (L \setminus u)^\omega\}$$

A step of reduction consists in the elimination of an element from $A(L)$, eventually compensated by the apparition of other elements. A first way to do this is described below:

Proposition 11. *Let $L \subseteq \Sigma^+$ be a rational language. For every $u \in A(L)$, both languages $G = u^*(L \setminus u)$ and especially $\Gamma = G \setminus GStab(L^\omega)$ are ω -generators of L^ω .*

Proof. Let $G = u^*(L \setminus u)$. As $u \in L$, we get that $G \subseteq L^+$ and then $G^\omega \subseteq L^\omega$. Conversely, let $w \in L^\omega$. There are two cases :

- either $w = u^\omega$ and then $w \in (L \setminus u)^\omega \subseteq (u^*(L \setminus u))^\omega$.
- either there exists $n \geq 0$ such that $w = xyw'$ where $x = u^n$, $y \in (L \setminus u)$ and $w' \in L^\omega$. Hence, $w \in (u^*(L \setminus u))L^\omega$. From Lemma 1, $w \in (u^*(L \setminus u))^\omega$.

The equality $L^\omega = G^\omega$ is proved. $L^\omega = \Gamma^\omega$ follows from Prop. 5. \square

Example 2. Let $L = a + ab + ba$. As (a, ba, ba, \dots) is a L -factorization of the word $(ab)^\omega$, we know that $A(L) = ab$. According to Prop. 11, the languages $G = (ab)^*(a + ba)$ and $\Gamma = a + (ab)^*ba$ are ω -generators of L^ω . Here, the latter language is an ω -code. It is necessarily a code and a reduced language too.

To increase the possibility to find a code when reducing a language, we have to treat separately the case where $A(L)$ contains two words sharing the same primitive root. So, here is a second way to remove an element from $A(L)$.

Proposition 12. *Let $L \subseteq \Sigma^+$ be a rational language. If $A(L)$ contains two non-empty words u and v such that u and v commute, then both languages $G = u + v^*(L \setminus \{u, v\})$ and especially $\Gamma = G \setminus GStab(L^\omega)$ are ω -generators of L^ω .*

Proof. From Lemma 2, $uv = vu$ implies that there exist two different integers $i \geq 1$ and $j \geq 1$ and a word $z \in \Sigma^+$ such that $u = z^i$ and $v = z^j$. As $\{z^i, z^j\} \subseteq A(L)$, according to Prop. 11, $G' = (z^j)^*(L \setminus z^j)$ is an ω -generator of L^ω . Moreover, $(z^j)^+z^i = z^i(z^j)^+ \subseteq G'Stab(L^\omega)$. Then, we obtain: $G' \setminus G'Stab(L^\omega) \subseteq G = z^i + (z^j)^*(L \setminus \{z^i, z^j\}) \subseteq G'$ and we deduce from Prop. 5 that G is an ω -generator of L^ω . From Prop. 5 again, $\Gamma = G \setminus GStab(L^\omega)$ is an ω -generator of L^ω . \square

Example 3. Let $L = a^2 + a^3 + b$. $A(L) = a^2 + a^3$ and we choose to remove a^3 . We deduce from Prop. 12 that $G = a^2 + (a^3)^*b$ and $\Gamma = a^2 + a^3b + b$ are ω -generators of L^ω . So Γ is an ω -code. The other choice would have lead to the ω -code $\Gamma' = a^3 + a^2b + a^4b + b$.

Obviously, the ω -generators computed by a step of reduction are not necessarily reduced. Perhaps the problem has just been moved. We study in the next section the use of the reduction, its range, and of course, its limit.

5.2 Experimentation

This section explains how to use the reduction in order to find reduced ω -generators, possibly codes. We limit ourselves to rational ω -powers which are adherences. Indeed, from Prop. 2 and Prop. 3, such ω -powers verify:

$$L^\omega = (\chi(L^\omega))^\omega = Adh(Root(\chi(L^\omega)))$$

Moreover, every finitely ω -generated language is an adherence [11]. From now on, L will denote the root of $\chi(L^\omega)$ which is ever characteristic of L^ω [8] and is, in addition here, the greatest ω -generator. The reduction principle consists in applying recursively either Proposition 11 or Proposition 12 to L and the languages obtained, while it is possible.

As an illustration, we make here a digression towards automata. Let \mathcal{A} be the minimal (deterministic) automaton which recognizes L^* , we note $L^* = L(\mathcal{A})$. Let \mathcal{B} be the same automaton in its Büchi version, L^ω is recognized by \mathcal{B} and we note $L^\omega = L_\omega(\mathcal{B})$. So, we intend to apply the reduction to L which is already minimal whenever it is not reduced. How does a reduction step operate on a deterministic Büchi automaton ? It suppresses a recognition state from one cycle. To do this, it induces a dilation of others as shown in Figure 1.

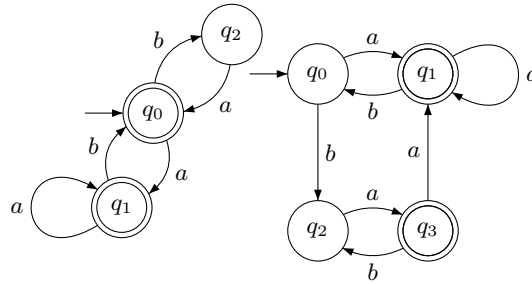


Fig. 1. Two automata for $L^* = (a + ab + ba)^*$ and $C^* = (a + (ab)^*ba)^*$ coupled with two Büchi automata for $L^\omega = C^\omega = (a + ab + ba)^\omega$.

Let us come back to the implementation of reduction. Thus, three different cases arise when we apply a step of reduction:

- (i) the process halts and gives an ω -generator code;
- (ii) the process halts on a reduced ω -generator which is not a code;
- (iii) the process does not halt.

In the sequel, we discuss the three cases from examples. The last two cases explore the actual limit of the reduction principle.

Case (i) In this case, the reduction provides a code, as illustrated in the following example. Note that the examples from Section 5.1 would have been convenient here. However, we give another example to show that, sometimes, it is also possible to get a code which is not ω -code.

Example 4. Let $L = ab + aba + baba$. The set $A(L) = ab$ since, in particular, $(aba, baba, baba, \dots)$ is a $(L \setminus ab)$ -factorization of the word $(ab)^\omega$. According to Prop. 11, the language $G = (ab)^*(aba + baba)$ is an ω -generator of L^ω . Hence, $\Gamma = aba + ababa + (ab)^*baba$ is a reduced ω -generator of L^ω . It is a code but not an ω -code. It was already known that L^ω has no ω -code as ω -generator [1].

Case (ii) This time, the process of reduction provides a reduced ω -generator which is not a code.

Example 5. Let $L = a + ab + bab$. We have $A(L) = ab$. According to Prop. 11, $\Gamma = a + bab + aba + ab^2ab$ is an ω -generator of L^ω . It is reduced but it is still not a code. However, there is no way to continue the reduction because Γ is reduced: one can easily verify that $A(\Gamma) = \emptyset$. In addition, L is a language with a bounded delay 1 and it was already known that L^ω cannot be ω -generated by an ω -code [1]. According to Prop. 4, L^ω has no code among its ω -generators.

Every time the process halts on a reduced ω -generator which is not a code, we succeed in finding a proof more or less *ad hoc* that the concerned ω -power is not ω -generated by a code. But examples are quickly difficult to handle and we do not know more. Nevertheless, it is not excluded that a generalization would be possible. We have to investigate for instance deciphering delays.

Case(iii) How do we interpretate the third case ? Our process does not halt. It clearly contains the case where L^ω has no reduced ω -generator, nor code.

Example 6. Let $L = a^2 + a^3 + ba + b$. We get $A(L) = a^2 + a^3$. According to Prop. 12, $\Gamma = a^2 + a^3ba + a^3b + ba + b$ and $\Gamma' = a^3 + a^2ba + a^2b + a^4ba + a^4b + ba + b$ are both ω -generators of L^ω . Neither Γ nor Γ' are reduced since $A(\Gamma) = a^3ba + a^3b$ and $A(\Gamma') = a^2ba + a^4ba + a^4b$. So neither Γ nor Γ' are codes. We can continue the process for a while ... But it necessarily continues without halting (nor looping) because we proved there is no reduced ω -generator for such an ω -language (in the proof of Prop. 9).

Is the condition sufficient ? We have no counterexample, nor proof. The only certitude is that the process cannot halt if there is no reduced languages among the ω -generators.

To decide if a rational adherence admits an ω -generator code, it is not sufficient to test whether the root of the greatest ω -generator is a code. The technique of reduction can help but obscure areas remain. Finally, several significative examples are recapitulated in Table 2 like in [19]. It is not worth exhibiting complicated examples to illustrate the complexity of the problem.

6 Conclusion

The research of ω -generators codes lead us to define the new class of reduced languages, strongly connected with periodic ω -words. Particularly, we have explained its remarkable position between code and minimal ω -generators though this area is not so large. In the rational case, the definition of reduced languages allows an algorithmic approach to search and sometimes find ω -generators codes. Up to what point does our method produce a code whenever it exists ? An intensive work of experimentation is needed to understand where is exactly the limit of such a method.

Table 2. Examples in brief.

$L =$ $\text{Root}(\chi(L^\omega))$	L^ω has an ω -generator ...		
	reduced	code	ω -code
$a + ab$	$a + ab$		
$a^2 + a^3 + b$	$a^2 + a^3b + b$		
$a + ab + ba$	$a + (ab)^*ba$		
$a + ab + b^2$	$a + ab + b^2$		no
$ab + aba + baba$	$aba + ababa + (ab)^*baba$		no
$a + ab + bc + c$	$a + ab + bc + c$		no
$a + ab + bab$	$a + bab + aba + ab^2ab$		no
$a^2 + a^3 + ba + b$	no		

References

1. X. Augros. Etude des générateurs de langages de mots infinis. Master's thesis, Univ. de Nice - Sophia Antipolis, june 1997.
2. J. Berstel and D. Perrin. *Theory of codes*. Academic Press, 1985.
3. L. Boasson and M. Nivat. Adherences of languages. *Journal of Computer and System Sciences*, 20:285–309, 1980.
4. J.R. Büchi. On a decision method in restricted second order arithmetics. In *International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, 1960.
5. J. Devolder. Generators with bounded deciphering delay for rational ω -languages. *Journal of Automata, Languages and Combinatorics*, 4(3):183–204, 1999.
6. J. Devolder, M. Latteux, I. Litovsky, and L. Staiger. Codes and infinite words. *Acta Cybernetica*, 11(4):241–256, 1994.
7. S. Julia. *Sur les codes et les ω -codes générateurs de langages de mots infinis*. PhD thesis, Université de Nice - Sophia Antipolis, 1996.
8. S. Julia. A characteristic language for rational ω -powers. In *Proc. 3rd Int. Conf. Developments in Language Theory*, pages 299–308, Thessaloniki, 1997.
9. S. Julia, I. Litovsky, and B. Patrou. On codes, ω -codes and ω -generators. *Information Processing Letters*, 60(1):1–5, oct. 1996.
10. J. Karhumäki. On three-element codes. *Theoret. Comput. Sc.*, 40:3–11, 1985.
11. M. Latteux and E. Timmerman. Finitely generated ω -languages. *Information Processing Letters*, 23:171–175, 1986.
12. I. Litovsky. Free submonoids and minimal ω -generators of R^ω . *Acta Cybernetica*, 10(1-2):35–43, 1991.
13. I. Litovsky. Prefix-free languages as ω -generators. *Information Processing Letters*, 37:61–65, 1991.
14. I. Litovsky and E. Timmerman. On generators of rational ω -power languages. *Theoretical Computer Science*, 53:187–200, 1987.
15. M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge, 2002.
16. D. Perrin and J.E. Pin. *Infinite words*. Elsevier Academic Press, 2004.
17. L. Staiger. On infinitary finite length codes. *Theoretical Informatics and Applications*, 20(4):483–494, 1986.
18. W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, volume B, chapter 4. Elsevier Science Publishers, 1990.
19. Tran Vinh Duc. A la recherche des codes générateurs de langages de mots infinis. Master's thesis, IFI Hanoi - Univ. de Nice - Sophia Antipolis, 2006.