International Journal of Computer and Information Sciences, Vol. 5, No. 2, 1976

Circular Codes and Synchronization

J.-L. Lassez[1]

Received June 1975; revised August 1975

Various characterizations of codes are given with a finite synchronization delay. Decidability and bounds on the delay are established in particular cases.

KEY WORDS: Codes; regular languages; synchronization.

1. INTRODUCTION

Considering words written as cycles,[7] we consider how the notion of coding fits in such a setting. The circular codes that we find are precisely the very pure codes of Ref. 6. Clearly the the noncircular codes leading to multiple decompositions of circular words have nonsynchronizing properties. This leads to the notion of parasite subwords (that is, a subdecomposition of a word, distinct from the main one), and to the classification of codes into three families: those with bounded parasitism (shown to be equivalent to codes with a finite synchronization delay), those with spread parasitism, and those with concentrated parasitism. It is known[6] that in the finite case very pure (and therefore circular) is equivalent to the existence of a finite synchronization delay, and a bound on the delay is found. This equivalence is not true in the regular case; however, we show that it remains true if we eliminate the simply defined family with concentrated parasitism. This allows us to find a decision procedure for finite synchronization delay in the regular case and a bound on this delay in a particular case.

Work done at Summer Research Institute of the Canadian Mathematical Congress, supported by the National Research Council of Canada.

[1] Département de Physique-Mathématiques, Université de Moncton, Moncton, Nouveau-Brunswick, Canada.

2. PRELIMINARIES

Let X be a nonempty finite set and let X^* be the free monoid generated by X. The elements of X will be called letters and those of X^* will be called *words*. A subset C of X^* is a *code* if and only if any word of C^* has a unique factorization in terms of elements of C; C^* is then a free submonoid of X^*. We know[2]:

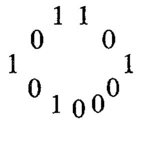
Proposition 1. A subset of C^* of X^* is a free submonoid if and only if $\forall u, v \in X^*$, uv and $vu \in C^* \Rightarrow u$ and $v \in C^*$ or u and $v \notin C^*$.

A code C is called *very pure*[5] if and only if $\forall u, v \in X^*$, uv and $vu \in C^* \Rightarrow u$ and $v \in C^*$. Let C be a code. A pair (u, v) of elements of C^* is *synchronizing* if and only if $\forall f, f' \in X^*$, $fuvf' \in C^*$ implies fu and $vf' \in C^*$. C has *synchronization delay* q if and only if every pair of elements of C^q is synchronizing. C has a *finite synchronization delay* if and only if it is synchronizing for some q.[6] We add precision to this definition: if for every u in C, $fuf' \in C^*$ implies f and f' belong to C^*, C has a synchronizing delay equal to 0. (Clearly such codes have stronger synchronizing properties than those with a delay equal to 1.) Let us recall the following basic facts from automata theory and regular languages. For any subset L of X^* we define the following equivalence relation: $\forall f, g \in X^*, f \equiv g(L)$ if and only if $\forall h \in X^*, fh \in L \Leftrightarrow gh \in L$. Clearly this is a right congruence for which L is saturated. L is regular if and only if the index of this relation is finite, in which case it is equal to the number of states of the minimal automaton accepting L.[1,3]

3. CIRCULAR CODES

We consider, following Rosenfeld,[7] words of X^* written in a circular way by bending and juxtaposition of the first and last letter of the word, as shown in the following example:

$X = \{0, 1\}$ $\qquad$ $01101000101 \rightarrow$



We shall say that a code is *circular* if and only if every circular word formed from a word of C^* admits a unique decomposition in terms of words of C. This loose definition may be clearly understood by looking at the two examples in Fig. 1. We have:

Theorem 1. A code is circular if and only if it is very pure.

Proof. We may note that if a code C is not circular, any circular word built from a word of C^* that has distinct decompositions in words of C is

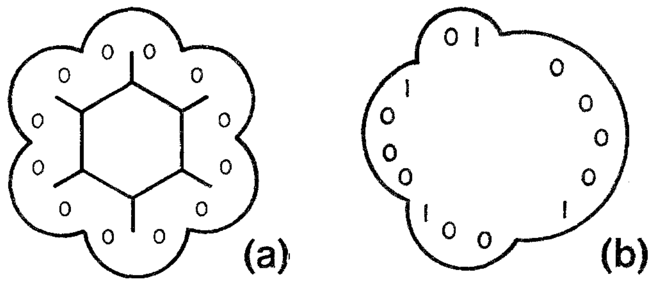


Fig. 1. (a) $C = (00)$ is not circular. (b) $C = (0^*1)$ is circular.

such that a parsing line of one of the decompositions is never in coincidence with a parsing line of another decomposition; otherwise we could break the circular word at this place and find a word of C^* having two distinct decompositions. Now by looking at the following picture, the demonstration comes easily: if a circular word has two distinct decompositions, let u be any word starting with a parsing line of one of the decompositions and ending with a parsing line of the other decomposition; and let v be the remaining part. If u is in C^*, by Proposition 1, v is also in C^*; this creates a third decomposition with parsing lines in coincidence with those of the two previous decompositions (see Fig. 2). Now if C is not very pure, we have two words u and v that do not belong to C^* such that uv and vu belong to C^*. From uv we build a circular word having two distinct decompositions, one starting with u and the other with v.

4. PARASITE SUBWORDS

If we break a circular word having two distinct decompositions in a way that respects one of the decompositions, as shown in Fig. 3, we find a word

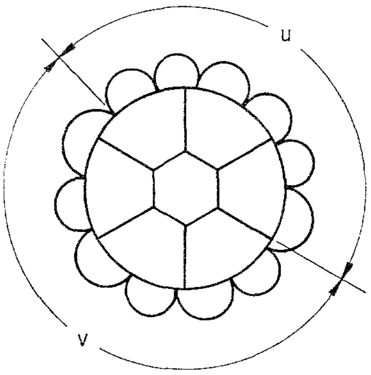


Fig. 2

Fig. 3

a of C^* that has another word b of C^* as a subword. As we saw previously, no parsing line of a is in coincidence with a parsing line of b; however b is not necessarily the largest subword of a in C^*, and eventually prefixes or suffixes of a may be in C^*. The following definition formalizes this fact:

Definition 1. Let C be a code, p and q positive integers, $a = a_1a_2 \cdots a_p$ ($a_i \in C$), $b = b_1b_2 \cdots b_q (b_i \in C)$; b will be called a *parasite subword* of a if and only if $\exists h, g \in X^*$ (h or $g \notin C^*$) such that $a = hbg$, and if $a_1a_2 \cdots a_i = hb_1b_2 \cdots b_j$, then $i = p$ and $j = q$.

C has *bounded parasitism* if and only if there exists an integer d such that no word of C^n, $n > d$, is a parasite subword of another word of C^*. The smallest such integer d will be called the *degree* of bounded parasitism.

A code C with unbounded parasitism has *spread parasitism* if and only if words of C cannot have parasite subwords in C^q with q arbitrarily large.

A code C has *concentrated parasitism* if and only if there exist words in C having parasite subwords in C^q with q arbitrarily large.

Clearly any code has one of these three mutually exclusive properties.

Now if a code has unbounded parasitism, trivially it cannot have a finite synchronization delay. Conversely if a code C does not have a finite synchronization delay, then $\forall q \exists u, v \in C^q, f, f' \in X^*$, $a = fuvf' \in C^*$, and fu or $vf' \notin C^*$. Let $a = a_1 \cdots a_n$, $u = u_1 \cdots u_q$, $v = v_1 \cdots v_q$ be the decompositions of a, u, v into words of C. Then for any integers i, j, k, l, $i < j, k < l$ we cannot have both equalities at the same time: $a_1 \cdots a_i = fu_1 \cdots u_k$ and $a_{j+1} \cdots a_n = v_l \cdots v_q f'$. Otherwise the word $a_{i+1} \cdots a_j = u_{k+1} \cdots v_{l-1}$ would have two distinct decompositions. So u or v is a parasite subword of a. We have therefore the simple:

Proposition 2. A code C has bounded parasitism if and only if it has a finite synchronization delay.

This proposition is clearly related to Theorem 3.1 of Restivo,[6] which we will use later on, reworded in our terminology:

Theorem 2. If a code has bounded parasitism of degree d, then it has a finite synchronization delay inferior or equal to d.

Now if a code is not very pure, $\exists u, v \notin C^*$ such that uv and vu belong to C^*, which implies double decompositions of circular words and directly the existence of parasite subwords. Now, using Proposition 1 twice, we see that for any integers n, p the following holds: $ab = (vu)^n v(u(vu)^n v)^p u \in C^*$ and $ba = (u(vu)^n v)^p u(vu)^n v \in C^*$ but $a = (vu)^n v$ and $b = (u(vu)^n v)^p u$ do not belong to C^*. Therefore we may have double decompositions involving an arbitrarily large number of elements in each decomposition. It follows immediately that a non-very pure code has unbounded parasitism, or:

Proposition 3. Every code with a finite synchronization delay is very pure.

The converse is not true in general; for instance, $C = \{0, 10^*2\}$ is circular (because if $uv \in C^*$ and $u \notin C^*$, then $vu \notin C^*$) and clearly without finite synchronization delay. However it is true in the finite case.[6] We will consider the regular case for which we need the following:

Lemma 1. A regular code C has concentrated parasitism if and only if $\exists h, g \in X^*, \exists a \in CC^*$ such that $ha^*g \subset C$.

Proof. The condition is sufficient because it follows clearly from the definition that for every n, a^n is a parasite subword of ha^ng. Conversely, we have words in C with parasite subwords in C^q, q arbitrarily large. Let $c, b_1, b_2, \ldots, b_q$ be words of C and $b_1b_2 \cdots b_q$ be a parasite subword of $c = hb_1 \cdots b_qg$. Regularity implies that there will be two integers i and $j (i < j)$ such that $hb_1 \cdots b_i \equiv hb_1 \cdots b_j\ (C)$; then $hb_1 \cdots b_i \equiv hb_1 \cdots b_i(b_{i+1} \cdots b_j)^n$ for any integer n. Therefore $hb_1 \cdots b_i(b_{i+1} \cdots b_j)^* b_{i+1} \cdots b_q g \subset C$.

Lemma 2. A regular code with spread parasitism is not very pure.

Proof. Let $b = b_1b_2 \cdots b_q\ (b_i \in C,\ 1 \leqslant i \leqslant q)$ be a parasite subword of $a = a_1 \cdots a_p = hb_1 \cdots b_q g\ (a_i \in C,\ 1 \leqslant i \leqslant p)$. Let h_i be the word associated to b_i such that $hb_1 \cdots b_i$ admits h_i as a suffix and h_i is a prefix of a_k for some k, such that $hb_1 \cdots b_i = a_1 \cdots a_{k-1}h_i$. Let us say that two such words h_i and h_j are equivalent if and only if they correspond to the same a_k, which implies that $b_{i+1} \cdots b_j$ is a subword of a_k. As q is arbitrarily large and spread parasitism implies that the difference between j and i is bounded, there will be an arbitrarily large number of equivalence classes. As C^* is regular, we may find two words h_i and h_j $(i < j)$ in two different classes such that $h_i \equiv h_j\ (C^*)$ (see Fig. 4). Let us define f as the word $b_{i+1} \cdots b_j = fh_j$. It

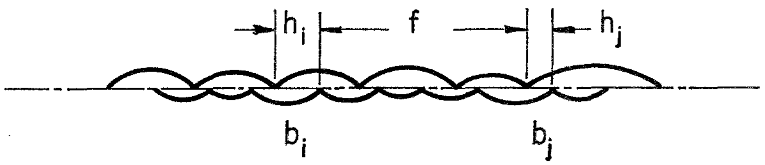


Fig. 4

is clear that $h_i f$ and fh_j both belong to C^*; now we have $h_i f \equiv h_j f(C^*)$ and, as $h_i f \in C^*$, then $h_j f \in C^*$. Therefore, using Proposition 1 and the fact that if h_j and f were in C^* then $fh_j = b_{i+1} \cdots b_j$ would have two distinct decompositions, we see that C is not very pure.

5. MAIN RESULTS

As a consequence of Lemma 2 and Propositions 2 and 3, we have the following:

Theorem 3. Let C be a regular code; then C has a finite synchronization delay if and only if C is very pure with no concentrated parasitism.

Theorem 4. It is decidable whether or not a regular code has a finite synchronization delay.

Proof. We verify first if C is very pure; if it is not, then it will not have a finite synchronization delay. This is decidable by considering the syntactic monoid M[4] associated with C^*: let H be the homomorphism from X^* onto M. As a direct consequence of the definitions, we see that C is very pure if and only if $\forall u, v \in X^*$, $H(u)\, H(v) \in H(C^*)$ and $H(v)\, H(u) \in C^*$ imply that $H(u)$ and $H(v)$ both belong to $H(C^*)$. As M is finite, this verification is always possible. Now if C is found to be very pure, we have to examine it for concentrated parasitism, that is (by Lemma 1), whether $\exists h, g \in X^*$ and $a \in CC^*$ such that $ha^*g \subset C$. This property is characteristic enough to be detected immediately in many cases; otherwise let s be a state of the minimal automaton accepting C, different from the dead state ($sX^* = s$) such that

$$\{m \in X^* \mid sm = s\} \cap C^*$$

is not reduced to the empty word; then C has concentrated parasitism. Conversely, if C has concentrated parasitism, it is clear that we will find such a state. So we have to verify whether the intersections of a finite number of regular languages with a given regular language are reduced to the empty word, which is decidable.

Now we give bounds on the synchronizing delay in a particular case. Let us recall that a *prefix code* is a set of words C such that no word of C is a strict prefix of another word of C.

Theorem 5. Let C be a regular prefix code. If C has a finite synchronization delay d, d is less than the number of states of the minimal automaton accepting C^*.

Proof. Proposition 2 tells us that C has bounded parasitism of degree n for some integer n, and Theorem 2 implies that n is an upper bound for d. Assume that n is equal to the number of states of the minimal automaton accepting C^*. Let $b = b_1 \cdots b_n$ $(b_i \in C, \quad 1 \leqslant i \leqslant n)$ be a parasite subword of $a = a_1 \cdots a_p = h_0 b_1 \cdots b_n g$, $(a_i \in C, \quad 1 \leqslant i \leqslant p)$. We can assume that a is the shortest sequence $a_k \cdots a_p$ having b as a parasite subword and therefore that h_0 is a prefix of a_1. As C is a prefix code and b is a parasite subword, $h_0 \notin C^*$. As in the proof of Lemma 2, let h_i $(1 \leqslant i \leqslant n)$ be the suffix of $h_0 b_1 \cdots b_i$ which is a prefix of a_k for some k; that is, $h_0 b_1 \cdots b_i = a_1 \cdots a_{k-1} h_i$. If h_i were reduced to the empty word, we would have $h_0 b_1 \cdots b_i = a_1 \cdots a_{k-1}$. By definition of h_i, $h_i = a_{k-1}$, this is impossible since the empty word cannot be an element of a code. Now, as we have $n + 1$ words h_i $(0 \leqslant i \leqslant n)$, two of them will be such that $h_i \equiv h_j (C^*)$ with $i < j$. If $1 \leqslant i < j \leqslant n$, h_i and h_j must be prefixes of the same a_k; otherwise we saw that it would imply that C is non–very pure, in contradiction with the assumption that C has a finite synchronization delay. Exactly the same proof holds if we allow $i = 0$. So h_i is a prefix of h_j, which is a prefix of a_k, with $0 \leqslant i < j \leqslant n$. Let $h_j = h_i b_{i+1} \cdots b_j$ and $a_k = h_i b_{i+1} \cdots b_j f$. For any integer m, we have

$$h_i \equiv h_i (b_{i+1} \cdots b_j)^m \, (C^*)$$

and, therefore, $h_i (b_{i+1} \cdots b_j)^m f \in C^*$. For m larger than n, $(b_{i+1} \cdots b_j)^m$ will not be a parasite sequence of $h_i (b_{i+1} \cdots b_j)^m f = c_1 \cdots c_q$ $(c_i \in C, 1 \leqslant i \leqslant q)$. Therefore we will find integers l, r, t $(r < q)$ such that

$$h_i (b_{i+1} \cdots b_j)^t \, b_{i+1} \cdots b_l = c_1 \cdots c_r \in C^*$$

So we have

$$x = h_i (b_{i+1} \cdots b_j)^t \, b_{i+1} \cdots b_l b_{l+1} \cdots b_j = h_i (b_{i+1} \cdots)^{t+1} \in CC^*$$

and, as $x \equiv h_i(C^*)$, h_i belongs to C^*.

Therefore $i \neq 0$ and, as C is a prefix code, we have $h_i = a_k$. This implies $h_0 b_1 \cdots b_i = a_1 \cdots a_{k-1} h_i = a_1 \cdots a_k$. By definition of a parasite sequence, we find $i = n$, in contradiction with $0 \leqslant i < j \leqslant n$. If no element of C^n is a parasite subword of an element of C^*, *a fortiori* no element of $C^m (m \geqslant n)$ will be, and the proof is concluded.

ADDENDUM

After this paper was submitted, the author was informed that Theorem 3 had been previously proved under a slightly different form by A. Restivo, which will appear in *Theoretical Computer Science.*

ACKNOWLEDGMENT

I wish to thank the referees for their many and valuable comments.

REFERENCES

1. A. Ginzgurg, *Algebraic Theory of Automata* (Academic Press, New York, 1968).
2. J.-L. Lassez, On the structure of systematic prefix codes, *Int. J. Comput. Math.*, Section A **3**, 177–188 (September 1972).
3. R. McNaughton and S. Papert, *Counter Free Automata* (MIT Press, Cambridge, Massachusetts, 1971).
4. R. McNaughton and S. Papert, "The Syntactic Monoid of a Regular Event," in *Algebraic Theory of Machines, Languages and Semigroups*, M. A. Arbib, ed. (Academic Press, New York, 1968), pp. 297–312.
5. A. Restivo, "Codes and Aperiodic Languages," in *Conference on Automata Theory and Formal Languages*, Bonn University, July 9–12, 1973.
6. A. Restivo, On a question of McNaughton and Papert, *Inf. Control* **25**(1), 93–101 (May 1974).
7. A. Rosenfeld, A note on cycle grammars, *Inf. Control* **27**(4), 374–377 (April 1975).

Printed in Belgium