A generalization of Girod's bidirectional decoding method to codes with a finite deciphering delay

L. Giambruno[1], S. Mantaci[2], J. Néraud[3], and C. Selmi[3]

[1] LIPN UMR CNRS 7030, Université Paris-Nord, 93430 Villetaneuse, France
[2] Dipartimento di Matematica e Informatica, Universitá di Palermo, 90133, Italy
[3] LITIS, Université de Rouen, 76801 Saint Etienne du Rouvray, France
giambruno@lipn.univ-paris13.fr, sabrina@math.unipa.it, Jean.Neraud@univ-rouen.fr, Carla.Selmi@univ-rouen.fr,

Abstract. In this paper we generalize an encoding method due to Girod (cf. [7]), based on binary prefix codes, that allows a bidirectional decoding of the encoded messages. In particular we generalize it to any finite alphabet A, to any operation defined on A, to any code with finite deciphering delay and to any key $x \in A^*$, whose length depends on the deciphering delay. We moreover define, as in [5], a deterministic transducer for such generalized method. We prove that, for a fixed code X over a given alphabet A with finite deciphering delay and for a fixed key $x \in A^*$, the transducers associated to different operations are isomorphic as unlabelled graphs. As an important result we moreover prove that, for a fixed code $X \subset A^+$ with finite deciphering delay, transducers associated to different keys have an isomorphic non trivial strongly connected component.

Introduction

Coding methods that allow bidirectional decoding of messages are used in order to guarantee data integrity. In fact, when we use a variable length code for source compression (cf. [10], Chapter 3), a single bit error in the transmission of the coded word may cause catastrophic consequences during decoding, since the wrongly decoded symbol generate loose of synchronization; in this way the error is propagated to the following symbols till the end of the file. In order to limit this error propagation, the compressed file is usually divided into records. The decoder scans the record in both directions. If just one error occurred in the coding, in this way it is possible to check the error position, isolate it and then avoid the error propagation. In order to do this we need codes that can be easily decoded in both directions. For this purpose, bifix codes are generally used, but usually either they are big, and then do not guarantee compression, or they are difficult to construct (cf. [4]). Prefix codes are usually very small instead, but is well known that, in spite they allow instantaneous decoding from left to right, the decoding from right to left can generally be performed with some deciphering delay. Due to a Schützenberger famous result (cf. [3], Chapter 3) such a delay is even infinite for maximal finite prefix codes that are not suffix. In 1999 B. Girod (cf. [7]) introduced a encoding/decoding method, that, even if it makes use of prefix codes, it allows an instantaneous decoding also from right to left by paying just few additional memory bits.

In previous papers (cf. [5], [6]) a bideterministic transducer is defined for the bidirectional deciphering of words by the method introduced by Girod [7]. Moreover an encoding method, inspired by the Girod's one, is introduced, and a transducer that allows both right-to-left and left-to-right decoding by this method is defined. Such a method depends on an alphabet, on a prefix code and on a word called key. It is proved also that this transducer is minimal.

Some bounds are also given on its number of states, depending on some features of the given code X.

In this paper we consider two different generalizations of the Girod's method. The first one concerns the cardinality of the code alphabet. This is realized by replacing the bitwise sum with a reversible operation defined by a *latin square map*, that is a map defined by a matrix of size $|A|^2$ and values on A, where no symbol is repeated on each row and each column. The second generalization concerns the decoding delay of the codes. As attested by the related literature, codes with a finite deciphering delay (f.d.d. in short) play a prominent part. Indeed, excepted infinite circular codes, the most best known classes of codes have f.d.d. Their topic is largely illustrated by deep powerful combinatorial results. We mention, among all, the two famous theorems of Schützenberger which concern the maximality of these codes [cf. [3], Chapter 5]. For this reason, it is natural to wonder what happens when, in a Girod's encoding/decoding suitable generalization, the prefix code is replaced by a code with some deciphering delay. As a consequence of our results, we can get a bidirectional deciphering delay equal to the minimal deciphering delay between the code and its reverse.

As another important result we show that the transducers associated to different keys have an isomorphic non trivial strongly connected component. This property emphasizes the deep connections between variable lengths codes and irreducible shifts (cf. [9]).

In Section 1 we introduce some preliminary notions on codes and transducers. In Section 2 we define latin square maps and we describe the generalization of Girod's method to f.d.d. codes. In Section 3 we give the algorithm for the construction of the deterministic transducer that realizes the decoding of bitstreams obtained by the Girod's encoding method from left to right and from right to left. We show that the transducers realizing left to right decoding and right to left decoding of the same inputs are isomorphic as graphs. We moreover show that the transducers over a given f.d.d. code $X \subset A^+$ and key $x \in A^*$, are isomorphic as unlabeled graphs, independently from the latin square maps used for encoding. In Section 4 we show that, for a given f.d.d. code X over a fixed alphabet A, the transducers associated to different keys have an isomorphic non trivial strongly connected component.

1 Preliminaries: codes and transducers

Let B and A be two alphabets, that we call respectively *source* alphabet and *channel* alphabet. Let $\gamma\colon B \to A^*$ be a map that associates to each element b in B a nonempty word over A. We extend this map to words over B by $\gamma(b_1 \dots b_n) = \gamma(b_1) \dots \gamma(b_n)$. We say that γ is an *encoding* if $\gamma(w) = \gamma(w')$ implies that $w = w'$. For each b in B, $\gamma(b)$ is said a *codeword* and the set of all codewords is said a *variable length code*, or simply a *code*. In what follows we denote by $x_i = \gamma(b_i)$ and by $X = \{x_1, \dots, x_m\}$ the code defined by γ. A *decoding* is the inverse operation than encoding i.e. the decoding of γ is the function γ^{-1} restricted to $\gamma(B^*)$. A set Y over A^* is said a *prefix set* (resp. *suffix set*) if no element of Y is a prefix (resp. a suffix) of another element of Y. Since one can prove that any prefix and any suffix set different from $\{\varepsilon\}$ is a code, we call them prefix and suffix codes, respectively. Words obtained by encoding a word in the source alphabet by a prefix code, can be decoded without delay in a left-to-right parsing.

Definition 1. *Let $X \subset A^*$ be a code and let d be a nonnegative integer. X is a code with finite deciphering delay (f.d.d. in short) d if for any $x, x' \in X$, $x \neq x'$, we have*

$$xX^dA^* \bigcap x'X^* = \emptyset.$$

The delay of X is the smallest integer d for which this property is verified. If such an integer does not exist we say that X has infinite deciphering delay.

Example 1. Any prefix code is a code with a f.d.d. $d = 0$. The code $X_d == \{01, (01)^d 1\} \subset A^*, A = \{0, 1\}$. is a code with f.d.d. d for any $d \geq 0$. In fact, $(01)X^d A^* \bigcap (01)^d 1X^* = \emptyset$. The code $X = \{0, 01, 11\} \subset A^*$ is a code with infinite deciphering delay. In fact, $0(11)^d 1 \in 0X^d A^* \bigcap 01X^*$ for all $d \geq 1$.

For each word $u \in A^*$, for each $k \leq |u|$, we denote by $pref_k(u)$ (resp. $suff_k(u)$) the prefix (resp. suffix) of u of length k and by $suff_{-k}(u)$ the suffix of u of length $|u| - k$. We denote by $\widetilde{u}$ the reverse of u. For $X = \{x_1, x_2, \ldots, x_n\}$, we define by $\widetilde{X}$ the set $\widetilde{X} = \{\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_n\}$. We denote by $Pref(X)$ (resp. $Suff(X)$) the set of prefixes (resp. suffixes) of words in X.

A finite *sequential transducer* T (cf. [11], [8] Chapter 1) is defined over an input alphabet A and an output alphabet B. It consists of a quadruple $T = (Q, i, F, \delta)$, where Q is a finite set of *states*, i is the unique *initial state*, δ is a partial function $Q \times A \longrightarrow B^* \times Q$ which breaks up into a *next state* function $Q \times A \longrightarrow Q$ and an output function $Q \times A \longrightarrow B^*$. The elements $(p, a, \delta(p, a)) = (p, a, v, q)$, with $p, q \in Q$, $a \in A$ and $v \in B^*$, are called *edges*. In this case we call a the *input label* and v the *output label*. F is a partial function $F : Q \longrightarrow B^*$ called the *terminal function*. We can represent encodings and decodings using transducers. In case of decoding, A represents the channel alphabet and B the source alphabet.

2 Generalization of Girod's method to f.d.d. codes

It is well known that a prefix code can be decoded without delay in a left-to-right parsing while it can not be as easily decoded from right to left. In [7] B. Girod introduced a very interesting alternative coding method using finite prefix codes on binary alphabets. It applies a transformation to a concatenation of codewords in a prefix code X that allows the deciphering of the coded word in both directions, by adding to the messages just as many bits as the length the longest word in X. This is somehow surprising, since, by paying a very small price, we obtain a bidirectional decoding of the encoded messages, even if the code is not necessarily suffix. We generalize this method to codes over alphabets with cardinality greater than two and with a f.d.d..

Let A be an alphabet and let f be a map from $A \times A$ into A. For all $a \in A$, we denote by $f_{(a,\cdot)}$ (resp. $f_{(\cdot,a)}$) the map from A into A defined by $f_{(a,\cdot)}(x) = f(a, x)$ (resp. $f_{(\cdot,a)}(x) = f(x, a)$), $\forall x \in A$. The maps $f_{(a,\cdot)}$ and $f_{(\cdot,a)}$ are called the *components* of f.

Definition 2. *A map $f : A \times A \longrightarrow A$ is said a* latin square map *on A if, for each $a \in A$, its components $f_{(a,\cdot)}$ and $f_{(\cdot,a)}$ are bijective.*

A latin square map on $A = \{0, \ldots, n\}$ is defined by a square array of size $n + 1$ where each line and each column contain one and only one occurrence of i, for all $0 \leq i \leq n$.

Example 2. This is an example of the latin square map g on $A = \{0, 1, 2\}$ defined by:

g	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

Remark 1. The bijectivity of the components of a latin square map f implies that there exists only one solution to the equation $f(a, b) = c$, in the case of one element among a, b or c is unknown. We can define two different "inverse" latin square maps associated to a given latin square map f. For $a, b, c \in A$ such that $f(a, b) = c$, we define the inverse maps f_1^{-1}, f_2^{-1} as $f_1^{-1}(c, b) = a$ and $f_2^{-1}(a, c) = b$. The corresponding square arrays can be easily computed in this way:

– for each $c, b \in A$, we define $a = f_1^{-1}(c, b)$ as the index of the row in the b-th column that contains the element c;
– for each $c, a \in A$, we define $b = f_2^{-1}(a, c)$ as the index of the column in the a-th row that contains the element c;

Let $X = \{x_1, \ldots, x_m\}$ be a finite code with f.d.d. d on an alphabet A, defined by an encoding γ over an alphabet $B = \{b_1, \ldots, b_m\}, x_j = \gamma(b_j), \forall\ 1 \leq j \leq m$. The words of X are named *codewords*. Let f be a latin square map on A. We extend f to the elements of $A^k \times A^k$, $k \geq 1$, by $f(a_1 \ldots a_k, a'_1 \ldots a'_k) = f(a_1, a'_1) \ldots f(a_k, a'_k)$. Let l be the length of the longest word in X and let x_L be a word in A^+ of length $L = (d+1)l$. Let $b = b_{i_1} \ldots b_{i_t} \in B^*$ and let $y = \gamma(b) = x_{i_1} \ldots x_{i_t}$, with $x_{i_j} \in X$, its encoding. Consider $y' = \widetilde{x}_{i_1} \ldots \widetilde{x}_{i_t}$ where $\widetilde{x}_{i_j}$ represents the reverse of x_{i_j}. Consider the word $z = f(yx_L, \widetilde{x}_L y')$. We define an encoding δ from B^* to A^* by $\delta(b) = z$. We can realize a left-to-right decoding of z by proceeding in the following way:

1. We first consider the words $v := \widetilde{x}_L$ and $t := z$, $t' := pref_L(t)$.
2. The word $u = f_1^{-1}(t', v)$ is a prefix of y with length L, then has a prefix which is product of at least $(d+1)$ codewords of X, that is $u = x_{i_1} \ldots x_{i_{d+1}} u'$, with $x_{i_j} \in X$, $u \in A^*$. Since X is a code with a f.d.d. d, we can state that the factorization of u begins with x_{i_1}. Therefore the decoding of z begins with b_{i_1}.
3. Let $v :=suff_L(v\widetilde{x}_{i_1})$ and let $t :=suff_{-|x_{i_1}|}(t)$ and $t' =pref_L(t)$. Coming back to step 2 we compute $u = f_1^{-1}(t', v)$, and since u has length L, we are able to recognize the first codeword of u, x_{i_2}. Then the decoding of z begins with $b_{i_1} b_{i_2}$.
4. This algorithm stops when $|t| = L$.

Example 3. This example gives an application of the previous decoding method. $X = \{01, 012\}$ is a code with f.d.d. $d = 1$ encoding $B = \{b_1, b_2\}$. We use for decoding the latin square map g of Example 2 and its inverse g_1^{-1} and we choose as key $x_L = 011011$ (then $\widetilde{x}_L = 110110$).
Let $y = (012)(01)(01)(012)(012)$ and $y' = (210)(10)(10)(210)(210)$. The encoding is performed by applying g to the pair $(yx_L, \widetilde{x}_L y')$

$$\begin{aligned} z &= g(0120101012012x_L, \widetilde{x_L}2101010210210) \\ &= g(012010101201211011, 110110210101021021 0) \\ &= 202200221100210020001 \end{aligned}$$

In order to decode z from left to right, we first consider $u = g_1^{-1}(202200, 110110) = 012010$. Since u has length 6, we are able to recognize the first codeword 012. Thus the decoding of z begins with b_2. After, we consider $v = 110210$ and $t' = 200221$ and $u = g_1^{-1}(110210, 200221) = 010101 = (01)0101$. Thus the decoding of z begins with $b_2 b_1$.
By proceeding this way we get the entire decoding of the message.

We remark that, by reversing the roles of yx_L and $\widetilde{x}_L y'$, we can decode the word form right to left, just by using the information that the first word ends with x_L and the inverse latin square map g_2^{-1} that allows to get b from c and a.

3 Transducers for decoding

Let $X = \{x_1, \ldots, x_m\} \in A^+$ be a code with f.d.d. d defined by an encoding γ over an alphabet $B = \{b_1, \ldots, b_m\}$. Let $x_L \in A^*$ with $|x_L| = L = (l+1)d$, where l is the length

of the longest word in X, and let f be a latin square map on A. For any sequence z of codewords in X, consider the encoding δ given by the generalization of Girod's method. The left-to-right decoding method given in the previous section can be described by the transducer, $\mathcal{T}(X)_{f,x_L} = (Q, i, \delta, F)$ defined as follows:

1. The states in Q are pairs of words (u, v) such that: **a)** $u \in Pref(X^{d+1})$; **b)** v is a a word in $Suff(\widetilde{x}_L)Suff(\widetilde{X}^{d+1})$ of length $L - |u|$.
2. The initial state $i = (\varepsilon, \widetilde{x}_L)$.
3. The function δ is defined as follows: **a)** $\delta((u, av), c) = (\varepsilon, (ub, v))$ if $b = f_1^{-1}(c, a)$ and ub is a proper prefix of X^{d+1}; **b)** $\delta((u, av), c) = (b_{i_1}, (x_{i_2} \dots x_{i_{d+1}}, v\widetilde{x}_{i_1}))$ if $b = f_1^{-1}(c, a)$ and $ub = x_{i_1}x_{i_2} \dots x_{i_{d+1}} \in X^{d+1}$. In all remaining cases the transitions are not defined.
4. F is defined only for the accessible states of the form (u, v), $u = x_1 \dots x_d \in X^d$, as the word $b_1 \dots b_d \in B^d$.

Example 4. $X = \{0, 01\}$ is a code with f.d.d. $d = 1$, encoding $B = \{b_1, b_2\}$. We use for decoding the latin square map g of the Example 2 and the key $x_L = 0101$. We obtain the following left-to-right decoding Girod's transducer associated to X:

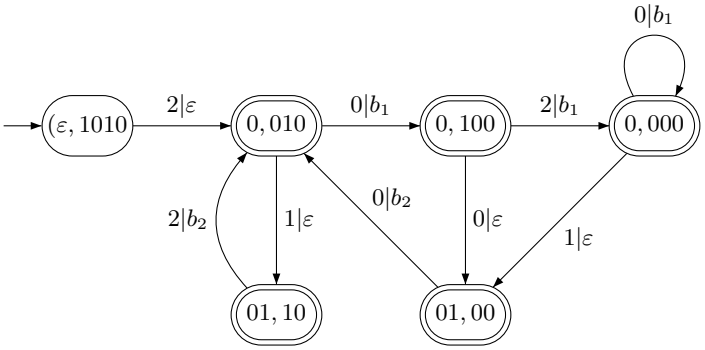

Fig. 1. Transducer T for the left-to-right decoding of $X = \{0, 01\}$ for $x_L = 0101$ over $B = \{b_1, b_2\}$

In a similar way we can define the transducer $\widetilde{\mathcal{T}}(X)_{f,x_L}$ for the right-to-left decoding. The following results hold:

Proposition 31 *The transducer $\mathcal{T}(X)_{f,x_L}$ (resp. $\widetilde{\mathcal{T}}(X)_{f,x_L}$) realizes the decoding δ^{-1} from left to right (resp. from right to left) on the encoded word z by reading the prefix (resp. the suffix) of length $|z| - L$ of z. Moreover this transducer is deterministic.*

Proposition 32 *The transducers $\mathcal{T}(X)_{f,x_L}$ and $\widetilde{\mathcal{T}}(X)_{f,x_L}$ are isomorphic as unlabelled graphs. If f is a commutative latin square map then they are also isomorphic as transducers.*

Proposition 33 *The transducers $\mathcal{T}(X)_{f,x_L}$ and $\mathcal{T}(X)_{g,x_L}$ are isomorphic as unlabelled graphs, for all pair of latin square maps f and g on A.*

4 A remarkable strongly connected component

In this section we examine the transducer $\mathcal{T}(X)_{f,x_L} = (Q, i, F, \delta)$ defined in the previous sections from the point view of graph theory. According to the results of Section 3, given two edges $((u, v), i, c, (u', v'))$, $((u, v), j, d, (u", v"))$ in the transducer $\mathcal{T}(X)_{f,x_L}$, if $i \neq j$, then $(u', v') \neq (u", v")$. This allows to introduce a graph $G(X)_{x_L}$, obtained by removing the labels from the transitions of $\mathcal{T}(X)_{f,x_L}$. According to the preceding construction:

Proposition 41 *Given a vertex (u, av) of $G(X)_{x_L}$ and $b \in A$, the following properties hold:*

1. *If ub is a proper prefix of X^{d+1} then a unique edge $(u, av) \to (ub, v)$ exists in $G(X)_{x_L}$.*
2. *If $ub = x_{i_1} \dots x_{i_{d+1}}$, with $x_{i_j} \in X (1 \leq j \leq d+1)$, then a unique edge $(u, av) \to (x_{i_2} \dots x_{i_{d+1}}, v\widetilde{x_{i_1}})$ exists in $G(X)_{x_L}$.*

Consider now $C(G(X)_{x_L})$, the subgraph of $G(X)_{x_L}$ where the vertices are elements of X^d $(Pref(X) \setminus X) \times Suff(\widetilde{X}^+)$, and where the edges are those in $G(X)_{x_L}$ connecting vertices in the subgraph. Remark that, as direct consequence of Proposition 41, given a vertex (u, v) of $C(G(X)_{x_L})$, any vertex accessible from (u, v) is a vertex of $C(G(X)_{x_L})$.

Lemma 42 *Let $(x_{i_1} \dots x_{i_d} p, v)$ be a vertex of $C(G(X)_{x_L})$, with $x_{i_j} \in X$, $\forall 1 \leq j \leq d)$, and let $z \in X$ such that p is a prefix of z. Then a path exists in $C(G(X)_{x_L})$ from $(x_{i_1} \dots x_{i_d} p, v)$ to $(x_{i_2} \dots x_{i_d} z, r\widetilde{x}_{i_1})$, where r is the suffix of v with length $|z| - |p|$.*

Proposition 43 *$C(G(X)_{x_L})$ is a strongly connected component of $G(X)_{x_L}$.*

As a consequence, we state the following result:

Proposition 44 *Given a code with a f.d.d. X, and given a key x_L, $C(G(X)_{x_L})$ is the unique non trivial strongly connected component of $G(X)_{x_L}$ which is accessible from any vertex of $G(X)_{x_L}$.*

As another remarkable property, the component $C(G(X)_{x_L})$ does not depend of the key x_L:

Theorem 45 *Given a code X with a f.d.d. d, a unique graph, namely $C(X)$ exists such that $C(X) = C(G(X)_{x_L})$, for any key x_L.*

References

1. M-P Béal, J. Berstel, B. H. Marcus, D. Perrin, C. Reutenauer and P. H. Siegel. Variable-length codes and finite automata. *In I. Woungang (ed), Selected Topics in Information and Coding Theory, World Scientific.* To appear.
2. J. Berstel and D. Perrin. *Theory of Codes.* Academic Press, 1985.
3. J. Berstel, D. Perrin and C. Reutenauer. *Codes and Automata.* Cambridge University Press, 2010.
4. A. S. Fraenkel and S. T. Klein. Bidirectional Huffman Coding, *The Computer Journal*, 33:296307.(1990)
5. L. Giambruno and S. Mantaci. Transducers for the bidirectional decoding of prefix codes, *Theoretical Computer Science*, 411:1785-1792.(2010)
6. L. Giambruno and S. Mantaci. On the size of transducers for bidirectional decoding of prefix codes. *Rairo-Theoretical Informatics and Applications*, DOI:10.1051/ita/2012006(2012)
7. B. Girod. Bidirectionally decodable streams of prefix code words. *IEEE Communications Letters*, 3(8):245–247, August 1999.
8. M. Lothaire. *Applied combinatorics on words*, Vol 104 of Encyclopedia of mathematics and its applications. Cambridge University Press, 2005.
9. D. Lind, B. Marcus. *An introduction to Symbolic Dynamics and Coding*, Cambridge University Press, 1995.
10. D. Salomon. *Variable-Length Codes for Data Compression.* Springer-Verlag, 2007.
11. J. Sakarovitch. *Éléments de théorie des automates.* Vuibert Informatique, 2003.