DOI: 10.1142/S0129054115500410

A Generalization of Girod's Bidirectional Decoding Method to Codes with a Finite Deciphering Delay

Laura Giambruno* and Sabrina Mantaci†
DMI, Universitá di Palermo
Palermo, Italy
**laura.giambruno@laposte.net*
†*sabrina@math.unipa.it*

Jean Néraud‡ and Carla Selmi§
LITIS, Université de Rouen
76801 Saint-Etienne de Rouvray, France
‡*jean.neraud@litislab.eu*
§*carla.selmi@litislab.eu*

Received 8 December 2013
Accepted 20 April 2015
Communicated by Jacques Sakarovitch

Girod's encoding method has been introduced in order to efficiently decode from both directions messages encoded by using finite prefix codes. In the present paper, we generalize this method to finite codes with a finite deciphering delay. In particular, we show that our decoding algorithm can be realized by a deterministic finite transducer. We also investigate some properties of the underlying unlabeled graph.

Keywords: Code; prefix (free) code; deciphering delay; transducer; unlabeled graph; strongly connected component.

Introduction

Coding methods that allow bidirectional decoding of messages (such as, for instance, bifix codes) assume a great importance in coding theory since they guarantee data integrity (cf. [9] Chapter 3). Unfortunately bifix codes are quite rare, difficult to be constructed and usually quite heavy with respect to the length of the codewords. On the other side, prefix free codes (for short prefix codes) are easy to be constructed, since they are in bijection with k-ary trees, and often very succinct. For instance Huffman codes, used in data compression, are prefix codes. Anyway their right to left decoding delay can be even infinite if the usual encoding method (each character is substituted to the corresponding codeword) is used.

In this context, Girod [6] introduced a very interesting alternative encoding method that, on one hand uses binary prefix codes, keeping in this way the succinctness property, and that, on the other hand, by the simple addition of few bits

to the encoded message (*encoding key*), allows an instantaneous bidirectional decoding of the encoded message. In [4] one can find a first study of the properties of the transducer defined in order to perform the bidirectional decoding Girod's algorithm. In [5] some statistics on the number of states of Girod's transducer are given. A very strong result for maximal codes due to Schützenberger (cf. [2]) states that a maximal finite code is either prefix or it has an infinite deciphering delay. This means that for maximal finite codes the only ones that can be decoded in both directions with a finite deciphering delay are bifix. Girod's encoding establishes a sort of symmetry in coding that allows, surprisingly, to decode in both directions maximal codes that are prefix and not suffix (i.e. have no deciphering delay from left to right and infinite deciphering delay from right to left) and symmetrically, codes, that are suffix and not prefix (for symmetrical reasons).

The present paper deals with the generalization of Girod's algorithm to the so-called codes with a finite deciphering delay (f.d.d. for short, cf. [2]), a natural extension of prefix codes: indeed prefix codes have left-to-right deciphering delay equal to zero. Furthermore, here we consider alphabets with an arbitrary cardinality and for this reason the bitwise sum used in Girod's encoding for two letters alphabets is substituted by more general operations defined by Latin squares. This new method, applied on a code X with a finite deciphering delay from left to right (resp. right to left), allows to decode a message in both directions.

Moreover an algorithm for decoding according to this method is implemented and realized by a deterministic transducer having remarkable properties. In particular, we prove that the underlying unlabeled graph contains an unique non trivial strongly connected component, that neither depends on the choice of the encoding key, nor on the Latin square but just on the code.

Section 1 contains the basic notions that we use in our paper. Some technical properties are stated in Remark 7. In Sec. 2, we describe our generalization of Girod's decoding method to f.d.d. codes. The corresponding decoding from right to left is implemented by a transducer described in Sec. 3. The corresponding decoding phase from right to left, is implemented by a similar transducer of which we describe the construction. Section 4 is devoted to investigating the properties of the above transducer and the corresponding graph. In Sec. 5 we give some conclusive remarks.

1. Preliminaries

1.1. *Codes*

In this section we give some notions and notation about codes used throughout the article. Let $B = \{b_1, \ldots, b_m\}$ and $A = \{0, \ldots, n\}$ be two alphabets, that we call respectively *source* alphabet and *channel* alphabet. Given an injective alphabetic morphism $\gamma\colon B^* \to A^*$, the set $X = \gamma(B) \subset A^*$ is called a *variable length code* or simply a *code*. The map γ is called an *encoding* and the words $x_i = \gamma(b_i)$ for all $1 \leq i \leq m$ are called *codewords*. The function γ^{-1} restricted to $\gamma(B^*)$ is the corresponding *decoding map*.

We recall that a *prefix free code* (or simply a *prefix code*) is a code where no codeword is a prefix of another one. In the present paper, we are interested in a special class of codes:

Definition 1. *Let $X \subset A^*$ be a code and let d be a non negative integer. X is a code with a (left-to-right)* finite deciphering delay (*f.d.d. code for 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 satisfied.*

Intuitively, messages encoded by f.d.d. codes can be decoded from left to right with a finite lookahead. The codes with $d = 0$ correspond to the class of prefix codes.

For each word u we denote by $\widetilde{u}$ the *mirror image* of u. For $X = \{x_1, x_2, \ldots, x_m\}$, we set $\widetilde{X} = \{\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_n\}$. Notice that if X is a code, $\widetilde{X}$ is a code as well.

Example 2. *Let $A = \{0, 1\}$.*

(1) *For any $d \geq 0$, let $X_d = \{01, (01)^d1\} \subset A^*$. The code X_d is a code with finite deciphering delay d. In fact, $(01)X^dA^* \bigcap (01)^d1X^* = \emptyset$. More intuitively, we cannot decide which one is the first element of a X-factorization of a prefix of the word $(01)^{d+1}1$ before having read the entire word.*

(2) *$X = \{0, 10, 11\} \subset A^*$ is a prefix code such that $\widetilde{X} = \{0, 01, 11\}$ is a code with an infinite deciphering delay. In fact, $0(11)^d1 \in 0X^dA^* \bigcap 01X^*$ for all $d \geq 1$.*

Given a set $X \subset A^+$ we denote by *Pref*(X) (resp. *Suff*(X)) the set of prefixes (resp. suffixes) of words in X. For each word $u \in A^*$ and for each $k \leq |u|$, we denote by *Pref*$_k(u)$ (resp. *Suff*$_k(u)$) the prefix (resp. suffix) of u having length k and by *Pref*(u) (resp. *Suff*(u)) the set of prefixes (resp. suffixes) of u.

1.2. *Transducers*

A finite transducer T, defined on an input alphabet A and an output alphabet B, consists of a quadruple $\mathcal{T} = \langle Q, I, E, T\rangle$ where Q is a finite set whose elements are called *states*, I and T are two distinguished subsets of Q called the sets of *initial and terminal states*, and E is a set of elements called *edges* which are quadruples (p, u, v, q) where p and q are states, u is a word over A and v is a word over B. We call u the *input label* and v the *output label*. An edge is commonly denoted by $p \xrightarrow{u|v} q$. Two edges $p \xrightarrow{u_1|v_1} q$ and $r \xrightarrow{u_2|v_2} s$ are *consecutive* if $q = r$. A *path* in a transducer is a sequence of consecutive edges. The *label of the path* is obtained by concatenating separately the input and the output labels. We denote it by a pair where the first element is a word over the input alphabet and second element is a word over the output alphabet. It defines a binary relation between words on the two alphabets as follows: a pair (u, v) is in the relation if it is the label of a successful

path (i.e. a path starting from the initial state i and finishing in a terminal state). This is called the *relation realized by* $\mathcal{T}$.

We can represent encoding and decoding algorithms using transducers. An encoding γ can be represented by a one-state literal transducer with loops on the state with labels $(b, \gamma(b))$, for each b in B. Transducers for decoding are more interesting. In case of decoding, A represents the channel alphabet and B the source alphabet.

A finite *sequential transducer* $\mathcal{T}$ (cf. [7] Chapter 1, [8] p. 578) over an input alphabet A and an output alphabet B consists of a quintuple $\mathcal{T} = \langle Q, i, \delta, T, F\rangle$, 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 $Q \times A \longrightarrow B^*$. T is the set of final states. In addition, $F : T \longrightarrow B^*$ is a partial function which is called the *terminal function*, by which an additional suffix can be attached to all the outputs.

1.3. *Latin squares*

We recall that the algorithm of Girod makes use of the modulo 2 sum [4]. In view of generalizing the method to larger alphabets, it is natural to consider modulo sum; however, we observe that when running, the corresponding algorithm does not make use of any underlying structure of group. Most general binary operations are defined by the so-called Latin squares.

Definition 3. *Let $A = \{0, \ldots, n\}$. A map $f : A \times A \longrightarrow A$ is a* Latin square *on A if and only if, for each $a \in A$, the mappings $x \longrightarrow f(a, x)$ and $x \longrightarrow f(x, a)$, which we call the components of f, are one-to-one.*

In other words a Latin square on A is $(n + 1) \times (n + 1)$ matrix where each row and each column contains one and only one occurrence of i, for all $0 \leq i \leq n$.

Example 4. *If $A = \{0, 1, 2\}$, the following matrix defines a Latin square:*

f	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

Since the components of a Latin square f are one-to-one maps, we can define two different "inverse" Latin squares associated to a given map f.

Definition 5. *We define the Latin squares f_r^{-1}, f_c^{-1}, respectively the row and column inverse maps, as $f_r^{-1}(f(a, b), b) = a$ and $f_c^{-1}(a, f(a, b)) = b$, for all $a, b \in A$.*

In view of simplifying the notation, if there is no ambiguity concerning the choice of the Latin square f, we shall set:

$$f(a, b) = a \oplus b,\ f_r^{-1}(a, b) = a \ominus b;\ f_c^{-1}(a, b) = a \oslash b.$$

Indeed, we shall see later another justification of such a notation (Corollary 16).

Remark 6. *The corresponding matrices can be easily computed in this way:*

- *For each $a, b \in A$, $a \ominus b$ is the index of the row in the b-th column that contains the value a;*
- *For each $a, b \in A$, $a \oslash b$ is the index of the column in the a-th row that contains the value b.*

Such a computation is illustrated in Example 8.

In a natural way, the preceding binary operations may be extended to words of arbitrary length. We summarize some elementary properties of a generic Latin square in the following remark:

Remark 7. *Let f be a Latin square:*

(1) $(u \oplus v) \ominus v = u$, *for all* $u, v \in A^*$ *such that* $|u| = |v|$.
(2) $u \oslash (u \oplus v) = v$, *for all* $u, v \in A^*$ *such that* $|u| = |v|$.
(3) $Pref_k(u \oplus v) = Pref_k(u) \oplus Pref_k(v)$, *for all* $u, v \in A^*$ *such that* $|u|, |v| \geq k$.
(4) $Suff_k(u \oplus v) = Suff_k(u) \oplus Suff_k(v)$, *for all* $u, v \in A^*$ *such that* $|u|, |v| \geq k$.

Latin squares are used in this context in order to generalize Girod's method to codes defined on alphabets of any cardinality. In fact in the original Girod's method, that only deal with binary codes, the binary sum is used, that can actually be defined by a very simple 2×2 Latin square, with zeros in the main diagonal and 1 elsewhere.

2. Extension of Girod's Method for f.d.d. Codes

In 1999 Girod (cf. [6]) introduced an efficient alternative coding method that uses finite prefix codes on binary alphabets. This method transforms a word obtained as concatenation of codewords from a prefix code X into a word which can be bidirectionally decoded without any delay, even if X is a not suffix code. This result is surprising, especially when the reverse code $\widetilde{X}$ is a code with an infinite deciphering delay (see for instance Example 2). The goal of this section is to generalize this method to codes with a finite deciphering delay over an alphabet with cardinality greater than two. This new method, applied on a code X with a finite delay from left to right or from right to left allows to decode a message in both directions. In other words, it is sufficient that the code has a finite deciphering delay from left to right or from right to left to obtain a decoding algorithm with finite deciphering delay in both directions. In the following we will say "deciphering delay" to mean "left-to-right dechiphering delay". This does not cause any loss of generality, since, in the case where the left-to-right dechiphering delay is infinite, the roles of X and $\widetilde{X}$ can be switched.

The generalized encoding algorithm. In the whole paper, a Latin square being fixed, we state the following notation:

- $X = \{x_1, \ldots, x_m\} \subset A^+$ stands for a finite code with finite deciphering delay d.
- γ is an injective alphabetic morphism $\gamma\colon B^* \to A^*$, where $B = \{b_1, \ldots, b_m\}$ and $x_j = \gamma(b_j)$, for all $1 \leq j \leq m$.
- We set $L = (d+1)\max\{[x]|x \in X\}$.
- We define the "encoding key" x_L as any given word in A^+ of length L.
- Let $y = x_{i_1} \ldots x_{i_t}, x_{i_j} \in X$, for $j = 1, \ldots, t$. We denote by $y' = \tilde{x}_{i_1}\tilde{x}_{i_2}...\tilde{x}_{i_t}$ the word obtained by concatenating the reverse, $\widetilde{x}_{i_j}$ of each codeword x_{i_j}, for $j = 1, \ldots, t$. Notice that y is different from $\widetilde{y}$.
- We denote by $\beta : B^* \longrightarrow A^*$ the map defined by $\beta(s) = \gamma(s)x_L \oplus \widetilde{x}_L\gamma(s)'$, for all $s \in B^*$.

Example 8. *Let $X = \{01, 012\}$ be a code with finite deciphering delay $d = 1$ encoding $B = \{b_1, b_2\}$ by a map γ. We consider the Latin square from Example* 4 *with its two corresponding inverses described as follows:*

$\oplus$	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

$\ominus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\oslash$	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

Let $x_L = 011011$ be the encoding key. Let $y = (012)(01)$ and let $y' = (210)(10)$. The encoding z of y is performed by applying the Latin square map to the pair $(yx_L, \widetilde{x}_L y')$

$$\begin{aligned}\beta(b_2b_1) &= 01201x_L \oplus \widetilde{x}_L 21010\\ &= 01201011011 \oplus 11011021010\\ &= 20220020001\,.\end{aligned}$$

In the proposition that we state below, we show that every $z \in \beta(B^*)$ is univocally decipherable in elements of B. In other words β is an encoding method that allows to generalize the classical encoding method from Girod.

Before proving this result, we introduce a transducer as defined in [[8] Chapter 5, p. 772]. This transducer, which we denote by $\mathcal{S} = \langle Q_{\mathcal{S}}, \delta_{\mathcal{S}}, i_{\mathcal{S}}, T_{\mathcal{S}}, F_{\mathcal{S}}\rangle$, generalizes to a code with a finite deciphering delay the classical flower transducer, which is used for decoding prefix codes. In the following figure, we give the transducer $\mathcal{S}$ for the left to right decoding of a text encoded by $X = \{01, 012\}$ (see Fig. 1).

Proposition 9. *Consider the notation given above. For each $z \in A^*$ there exists at most one element $s \in B^*$ such that $z = \beta(s)$.*

Proof. By contradiction, we suppose that there exist two different words $s, \overline{s} \in B^*$ such that $z = \beta(s) = \beta(\overline{s})$. Let us denote $y = \gamma(s)$ and $\overline{y} = \gamma(\overline{s})$, then we get

$$(1)\quad yx_L \oplus \widetilde{x}_L y' = \overline{y}x_L \oplus \widetilde{x}_L \overline{y}'$$

and $|y| = |\overline{y}|$. Since γ is an injective map, it suffices to prove that $y = \overline{y}$.

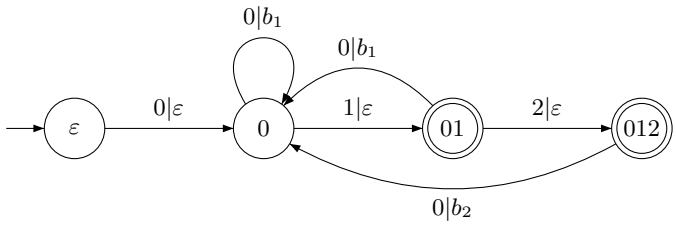

Fig. 1. The transducer $\mathcal{S}$ for a text encoded by $X = \{01, 012\}$ with $F_{\mathcal{S}}(01) = b_1$ and $F_{\mathcal{S}}(012) = b_2$.

First, we assume that $|y| \leq L$. By Remark 7 we get $z = y \oplus Pref_{|y|}(\widetilde{x}_L) = \overline{y} \oplus Pref_{|y|}(\widetilde{x}_L)$ and by the injectivity of the components of the Latin square map, it follows that $y = \overline{y}$.

Now, let $|y| > L$. Then y and $\overline{y}$ are products of at least $d+1$ elements of X. We set:

$$y = x_{i_1} \cdots x_{i_r} x_{i_{r+1}} \cdots x_{i_{r+d+1}}, \text{ with } x_{i_j} \in X, \text{ for all } \ 1 \leq j \leq r+d+1$$
$$\overline{y} = \overline{x}_{i_1} \cdots \overline{x}_{i_t} \overline{x}_{i_{t+1}} \cdots \overline{x}_{i_{t+d+1}}, \text{ with } \overline{x}_{i_j} \in X, \text{ for all } \ 1 \leq j \leq t+d+1\,.$$

In what follows we prove, by iteration on the integer r, that $\overline{y} \in x_{i_1} \ldots x_{i_r} A^*$.

Let $u_1 = Pref_L(z) \ominus \widetilde{x}_L$. By Remark 7 and since $|y| > L$ we obtain that u_1 is the prefix of length L of y. For the same reason u_1 is the prefix of length L of $\overline{y}$. Since $|u_1| = L$ and since X is a code with finite deciphering delay d, there exists a unique $x_{i_1} \in X$ such that $u_1 \in x_{i_1} A^*$. In what follows we indicate how to compute x_{i_1} by applying the transducer $\mathcal{S}$ introduced above: Let u_1' be the shortest prefix of u_1 such that $\delta_{\mathcal{S}}(i_{\mathcal{S}}, u_1') = (q_{i_1}, b_{i_1}), q_{i_1} \in T_{\mathcal{S}}, b_{i_1} \in B$. We have $x_{i_1} = \gamma(b_{i_1})$. Then both y and $\overline{y}$ begin with x_{i_1}. Let $z' = Suff_{|z|-|x_{i_1}|}(z)$ and $u_2 = Pref_L(z') \ominus Suff_L(\widetilde{x}_L \widetilde{x}_{i_1})$. As for u_1, we have $u_2 = Pref_L(x_{i_1}^{-1} y)$ (in this notation, $u^{-1}v$ stands for the unique word w such that $uw = v$). Therefore, we obtain $u_2 \in x_{i_2} A^*$. The computation of x_{i_2} will be done in a way similar to the computation of x_{i_1}: let u_2' be the shortest prefix of u_2 such that $\delta_{\mathcal{S}}(q_{i_1}, u_2') = (q_{i_2}, b_{i_2}), q_{i_2} \in T_{\mathcal{S}}, b_{i_2} \in B$; we have $x_{i_2} = \gamma(b_{i_2})$. Then $y \in x_{i_1} x_{i_2} A^*$ and, analogously, $\overline{y} \in x_{i_1} x_{i_2} A^*$. By iterating this process, we get $y, \overline{y} \in x_{i_1} \ldots x_{i_r} A^*$. Each codeword x_{i_j} is computed as indicate above.

Now, we set:

$$y_r = x_{i_{r+1}} \cdots x_{i_{r+d+1}} \text{ and } \overline{y}_r = \overline{x}_{i_{r+1}} \cdots \overline{x}_{i_{t+d+1}}\,.$$

Let us prove now that $y_r = \overline{y}_r$. According to (1), since $x_{i_j} = \overline{x}_{i_j}, \ \forall 1 \leq j \leq r$, we have:

$$y_r x_L \oplus Suff_{|y_r|+L}(\widetilde{x}_L y') = \overline{y}_r x_L \oplus Suff_{|y_r|+L}(\widetilde{x}_L \overline{y}').$$

Thus, according to Statement 4 of Remark 7, we obtain:

$$Suff_{|y_r|+L}(y_r x_L \oplus \widetilde{x}_L y') = Suff_{|y_r|+L}(\overline{y}_r x_L \oplus \widetilde{x}_L \overline{y}').$$

By considering the suffix of length L, we have:

$$x_L \oplus \mathit{Suff}_L(y') = x_L \oplus \mathit{Suff}_L(\overline{y}').$$

From the injectivity of the Latin square, we obtain $\mathit{Suff}(y') = \mathit{Suff}_L(\overline{y}')$. Since $|y'_r| = |\overline{y}'_r| \leq L, y'_r \in \mathit{Suff}(y)$ and $\overline{y}'_r \in \mathit{Suff}(\overline{y})$, we have $y'_r = \overline{y}'$. It follows $y_r = \overline{y}_r$, and this completes the proof. □

The proof of Proposition 9 suggests the implementation of the following algorithm for the left-to-right decoding. Let $z = \beta(s)$, where s is the concatenation of at least $d+1$ elements:

(1) Set the variables $v := \widetilde{x}_L$, $t := z$ and $t' := \mathit{Pref}_L(t)$.
(2) The word $u = t' \ominus v$ is the product of at least $(d+1)$ codewords of X, that is $u = x_{i_1} \dots x_{i_{d+1}} u'$, with $u' \in A^*$ and $x_{i_j} \in X$ for all $1 \leq j \leq d+1$. Since X is a code with finite deciphering delay d, the factorization of u begins by x_{i_1}. Therefore the decoding of z begins by b_{i_1}.
(3) Set $v := \mathit{Suff}_L(v\widetilde{x}_{i_1})$, $t := \mathit{Suff}_{|t|-|x_{i_1}|}(t)$, $t' = \mathit{Pref}_L(t)$.
In order to get all the factors of the coding, iterate Step 2 followed by Step 3 until $t' = t = x_L$ ($|t'| = |t| = L$).

Example 10. [Example 8 (continued)] *This example gives an application of the previous decoding method on* $z = 20220020001$ *as defined in the Example* 8.

In order to decode z *from left to right, we consider* $L = 6$ *and* $t' = \mathit{Pref}_6(z) = 202200$ *and*

$$\begin{aligned} u &= t' \ominus \widetilde{x}_L \\ &= 202200 \ominus 110110 \\ &= 012010. \end{aligned}$$

Since $|u| = L$, *we are able to recognize the first codeword* 012. *Thus the decoding of* z *begins with* b_2.

Afterwards, we consider $v = \mathit{Suff}_L(\widetilde{x_L 012}) = 110210$, $t = \mathit{Suff}_{|z|-|012|}(z) = 20020001$ *and* $t' = \mathit{Pref}_6(t) = 200200$. *We have:*

$$\begin{aligned} u &= t' \ominus v \\ &= 200200 \ominus 110210 \\ &= 010110. \end{aligned}$$

Since u *is prefix of* $u' = t \ominus v = 01011011 \in X^d x_L$ *we are able to recognize the first codeword* 01. *Thus the decoding of* z *begins with* $b_2 b_1$. *Since the length of the remaining part of* z *is exactly* 6, *this means that* $b_2 b_1$ *is the decoding of* z.

Similar arguments on the reversed words lead to implement a corresponding right-to-left decoding algorithm. In particular, the input word $\widetilde{x}_L y'$ is substituted to $y x_L$, the inverse Latin square $\oslash$ is substituted to $\ominus$.

In the previous algorithm, Step (2) may be completed by reading the word u in the transducer $\mathcal{S}$ introduced above. In what follows, we indicate a more precise scheme of this method:

Algorithm
{Input $z \in A^*$ Output $s = \beta^{-1}(z)$}
$u \leftarrow \varepsilon$; $v \leftarrow \widetilde{x}_L$; $t \leftarrow z$; $q \leftarrow i_{\mathcal{S}}$;
$s \leftarrow \varepsilon$; $x \leftarrow \varepsilon$; $b \leftarrow \varepsilon$;
while $|t| > L$ **do**
  $a \leftarrow Pref_1(t) \ominus Pref_1(v)$;
  $u \leftarrow ua$;
  $(q, b) \leftarrow \delta_{\mathcal{S}}(q, a)$;
  $\{b \in B \cup \{\varepsilon\}\}$
  if $q \notin T_{\mathcal{S}}$ **then**
    $\{u \in Pref(X^{d+1}) \setminus X^{d+1}, b = \varepsilon\}$
    $v \leftarrow A^{-1}v$; $t \leftarrow A^{-1}t$
  endif
  if $q \in T_{\mathcal{S}}$ **then**
    $\{u \in X^{d+1}, u = xX^d, x \in X, b \in B\}$
    $x \leftarrow \gamma(b)$;
    $u \leftarrow x^{-1}u$;
    $v \leftarrow v\widetilde{x}$; { Step (a)}
  endif
  if $q = \emptyset$ **then**
    {*z is not an encoding of Girod* }
    exit
  endif
endwhile;
if $q \in X^{d+1}$ **then**
  $s \leftarrow sF_{\mathcal{S}}(q)$
endif
endalgorithm

In Step (a), the computation of v, makes necessarily use of a memory for stocking x. Taking account of this remark, an alternative implementation of this algorithm consists in computing, in a preprocessing phase, the transducer $\mathcal{T}_{f,x_L}$ as indicated in Sec. 3. Actually, in this transducer, the states are the pairs (u, v) defined in the previous algorithm. After having constructed this transducer, the processing phase consists only in reading the input word on line. No more operation then applying output function, being required.

3. Transducers for Decoding

In this section, we prove that two previous decoding algorithms (for the left to right and the right to left decoding) can be realized by some particular transducers.

Transducer for left-to-right decoding. The left-to-right decoding algorithm given in Sec. 2 can be described by the transducer $\mathcal{T}(X)_{f,x_L} = \langle Q, i, \delta, T, F\rangle$ defined as follows:

(1) Q contains pairs of words (u, v) such that:
- $u \in Pref(X^{d+1}) \setminus X^{d+1}$;
- $v \in \bigcup_{i=0}^{d+1} Suff(\widetilde{x}_L \widetilde{X}^i)$ of length $L - |u|$.

(2) The initial state is $i = (\varepsilon, \widetilde{x}_L)$.

(3) For each state (u, av), for each $c \in A$, let $b = f_r^{-1}(c, a)$, the transition function δ is defined as follows:

$$\delta((u, av), c) = \begin{cases} (\varepsilon, (ub, v)), & \text{if } ub \in Pref(X^{d+1}) \setminus X^{d+1}; \\ (b_{i_1}, (x_{i_2} \dots x_{i_{d+1}}, v\widetilde{x}_{i_1})), & \text{if } ub = x_{i_1} \dots x_{i_{d+1}} \in X^{d+1}; \\ undefined, & \text{otherwise.} \end{cases}$$

(4) The set of the final states is $T = \{(u, v) | u = x_{i_1} \dots x_{i_d} \in X^d\}$.

(5) The function F is defined only for the accessible final states (u, v), $u = x_{i_1} \dots x_{i_d} \in X^d$, as the word $b_{i_1} \dots b_{i_d} \in B^d$.

Example 11. *$X = \{0, 01\}$ is a code with finite deciphering delay $d = 1$ defined by an encoding over $B = \{b_1, b_2\}$. We use for decoding the Latin square inverse $\ominus$ given in Example* 8 *and the key $x_L = 0101$. We obtain the following transducer associated to X for decoding from left to right. The output function F is defined for final states as : $F(0, 010) = b_1$, $F(0, 100) = b_1$, $F(0, 000) = b_1$, $F(01, 10) = b_2$, $F(01, 00) = b_2$.*

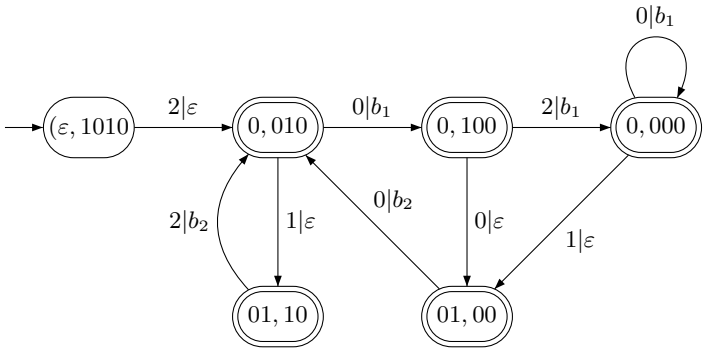

Fig. 2. The transducer $\mathcal{T}$ for the left-to-right decoding of a text encoded by $X = \{0, 01\}$, using encoding key $x_L = 0101$ over $B = \{b_1, b_2\}$ with Latin square f of the Example 4.

Transducer for right-to-left decoding. In a similar way, we can define the transducer $\widetilde{\mathcal{T}}(X)_{f,x_L} = \langle \widetilde{Q}, \widetilde{i}, \widetilde{\delta}, \widetilde{T}, \widetilde{F}\rangle$ for the right-to-left decoding as follows:

(1) $\widetilde{Q}$, contains pairs of words (u, v) such that:
- $u \in Suff(\widetilde{X}^{d+1}) \setminus \widetilde{X}^{d+1}$;
- $v \in \bigcup_{i=0}^{d+1} X^i Pref(x_L)$ of length $L - |u|$.

(2) The initial state is $\widetilde{i} = (\varepsilon, x_L)$.
(3) For each state (u, va), for each $c \in A$, let $b = f_c^{-1}(a, c)$, then the transition function δ is defined as follows:

$$\widetilde{\delta}((u,va),c) = \begin{cases} (\varepsilon,(bu,v)), & \text{if } bu \in Suff(\widetilde{X}^{d+1}) \setminus \widetilde{X}^{d+1}; \\ (b_{i_1},(\widetilde{x}_{i_2}\dots\widetilde{x}_{i_d},x_{i_1}v)), & \text{if } bu = \widetilde{x}_{i_1}\dots\widetilde{x}_{i_{d+1}} \in \widetilde{X}^{d+1}; \\ undefined, & \text{otherwise.} \end{cases}$$

(4) The set of the final states is $\widetilde{T} = \{(u,v) | u = \widetilde{x}_{i_1}\dots\widetilde{x}_{i_d} \in \widetilde{X}^d\}$.
(5) $\widetilde{F}$ is defined only for the accessible final states (u,v), $u = \widetilde{x}_{i_1}\dots\widetilde{x}_{i_d} \in \widetilde{X}^d$, as the word $b_{i_1}\dots b_{i_d} \in B^d$.

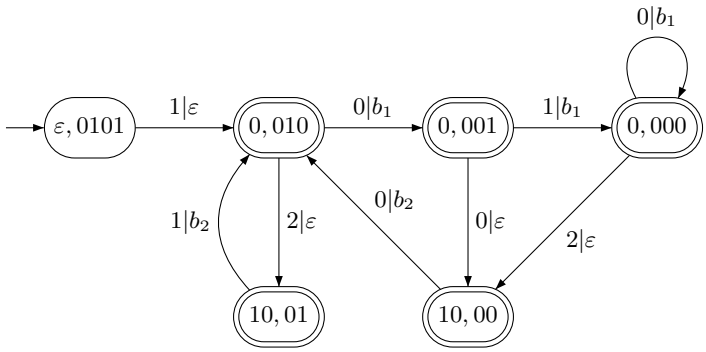

Fig. 3. The transducer $\tilde{\mathcal{T}}$ for the right-to-left decoding of $X = \{0, 01\}$ for $x_L = 0101$ over $B = \{b_1, b_2\}$ with the Latin square f of the Example 4.

Example 12. *$X = \{0, 01\}$ is a code with finite deciphering delay $d = 1$ defined by an encoding over $B = \{b_1, b_2\}$. We use for decoding the Latin square inverse $\oslash$ given in Example* 8 *and the encoding key $x_L = 0101$. We obtain the following transducer associated to X for decoding from right to left. The output function $\widetilde{F}$ is defined for final states as : $\widetilde{F}(0,010) = b_1$, $\widetilde{F}(0,001) = b_1$, $\widetilde{F}(0,000) = b_1$, $\widetilde{F}(10,10) = b_2$, $\widetilde{F}(10,00) = b_2$.*

As one can notice from Figs. 2 and 3, the transducers $\mathcal{T}(X)_{f,x_L}$ and $\widetilde{\mathcal{T}}(X)_{f,x_L}$ are isomorphic as unlabeled graphs. This is a general property proved in the next section.

The proof that the transducer $\mathcal{T}(X)_{f,x_L}$ (resp. $\widetilde{\mathcal{T}}(X)_{f,x_L}$) really realizes the left-to-right decoding (resp. right-to-left decoding) lays upon the results of Lemma 13 and Proposition 14. Since the proofs of these two statements are straightforward but technical, we suggest to the reader, that he (she) at first report in Sec. 4 to continue the reading of the paper. decoding (resp. right-to-left decoding), stated in Proposition 14, we first establish the following Lemma.

Lemma 13. *Transducer $\mathcal{T}(X)_{f,x_L}$ contains a path from the initial state $(\varepsilon, \widetilde{x}_L)$ to a final state (u,v) labeled $(z, b_{i_1}\dots b_{i_k}), k \geq 0$, if and only if*

(1) $z = x_{i_1} \dots x_{i_k} u \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_k}|+|u|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k})$,
(2) $v = \mathit{Suff}_{L-|u|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k})$.

Proof. Let us consider a successful path from $(\varepsilon, \widetilde{x}_L)$ to a final state (u, v) with label $(z, b_{i_1} \dots b_{i_k})$. We prove the first direction by induction on $k \geq 0$.

If $k = 0$ then we have a path from $(\varepsilon, \widetilde{x}_L)$ to a final state (u, v) with label (z, ε) and, by construction $u = x_{i_1} \dots x_{i_d}$, $v = \mathit{Suff}_{L-|u|}(\widetilde{x}_L)$ and $z = u \oplus \mathit{Pref}_{|u|}(\widetilde{x}_L)$.

Let us consider now $k > 0$ and let

$$(\varepsilon, \widetilde{x}_L) \overset{z'|b_{i_1} \dots b_{i_{k-1}}}{\longrightarrow} (u', v') \overset{z''|b_{i_k}}{\longrightarrow} (u'', v'')$$

be a successful path. By induction, we have:

$$\begin{aligned} &u' = x_{i_k} \dots x_{i_{d+k-1}}, \\ &v' = \mathit{Suff}_{L-|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}), \\ &z' = x_{i_1} \dots x_{i_{k-1}} u' \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}). \end{aligned}$$

First, we note that, (1) $u'' = x_{i_k}^{-1} u' x_{i_{d+k}} = x_{i_{k+1}} \dots x_{i_{d+k}}$, hence, we have (2) $|u''| - |u'| = |x_{i_{d+k}}| - |x_{i_k}|$. Therefore, since $|u''| + |v''| = |u'| + |v'| = L$, we obtain (3) $|v''| + |x_{i_{d+k}}| = |v'| + |x_{i_k}|$. We have also that $z'' = x_{i_{d+k}} \oplus \mathit{Pref}_{|x_{i_{d+k}}|}(v')$.

Since $u' = x_{i_k} \dots x_{i_{d+k-1}}$ and, by (1), $x_{i_1} \dots x_{i_{d+k}} = x_{i_1} \dots x_{i_k} u''$, we have that:

$$\begin{aligned} z &= (x_{i_1} \dots x_{i_{d+k-1}} \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}))(x_{i_{d+k}} \oplus \mathit{Pref}_{|x_{i_{d+k}}|}(v')) \\ &= x_{i_1} \dots x_{i_{d+k-1}} x_{i_{d+k}} \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}) \mathit{Pref}_{|x_{i_{d+k}}|}(v') \\ &= x_{i_1} \dots x_{i_k} u'' \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}) \mathit{Pref}_{|x_{i_{d+k}}|}(v'). \end{aligned}$$

Now, we compute $r = \mathit{Pref}_{|\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}) \mathit{Pref}_{|x_{i_{d+k}}|}(v')$:

$$\begin{aligned} r &= \mathit{Pref}_{|\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}) \mathit{Pref}_{|x_{i_{d+k}}|}(\mathit{Suff}_{L-|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}})) \\ &= \mathit{Pref}_{|\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}) \mathit{Pref}_{|x_{i_{d+k}}|}(\mathit{Suff}_{L-|u'|+|\widetilde{x}_{i_k}|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k})). \end{aligned}$$

Set $t' = \mathit{Pref}_{|\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}|+|u'|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}})$ and $t = \mathit{Suff}_y L - |u' + |\widetilde{x}_{i_k}|(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}})$.

We have $|t'|+|t| = |\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}|+|u'|+L-|u'|+|\widetilde{x}_{i_k}| = |\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k}|$. It follows that $t't = \widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k}$, whence $r = t' \mathit{Pref}_{|x_{d+k}|}(t) = \mathit{Pref}_{|z|+|x_{d+k}|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k})$. We compute $|r| = |t'| + |x_{i_{d+k}}| = |\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}| + |u'| + |x_{i_{d+k}}|$. By (3), we have:

$$\begin{aligned} |r| &= |\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}| + |u'| + |v'| - |v''| + |\widetilde{x}_{i_k}| \\ &= |\widetilde{x}_{i_1} \dots \widetilde{x}_{i_{k-1}}| + L - |v''| + |\widetilde{x}_{i_k}| \\ &= |\widetilde{x}_{i_1} \dots \widetilde{x}_{i_k}| + |u''|. \end{aligned}$$

As a consequence, we obtain $r = \mathit{Pref}_{|\widetilde{x}_{i_1} \dots \widetilde{x}_{i_k}|+|u''|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k})$. This implies that $z = x_{i_1} \dots x_{i_k} u \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_k}|+|u''|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k})$ and this completes the proof.

The reverse direction follows by remarking that every previous statement is invertible. □

As a consequence of Lemma 13, the following result holds:

Proposition 14. *With the preceding notation, the pair* $(z, s) \in A^* \times B^*$, *with* $s = b_{i_1} \dots b_{i_k}, k \geq 0$, *is the label of a successful path in* $\mathcal{T}(X)_{f,x_L}$ *ending at* (u, v) *if and only if* z *is the prefix with length* $|x_{i_1} \dots x_{i_k} u|$ *of the word* $\beta(s)$.

Proof. Let $(z, s) \in A^* \times B^*$ be the label of a successful path in $\mathcal{T}(X)_{f,x_L}$. According to Lemma 13, we have

$$z = x_{i_1} \dots x_{i_k} u \oplus \mathit{Pref}_{|x_{i_1} \dots x_{i_k}|+|u|}(\widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k}),$$

with $u = x_{i_{k+1}} \dots x_{i_{k+d}}$. By Remark 7, Property 3, we obtain

$$\begin{aligned} z &= \mathit{Pref}_{|x_{i_1} \dots x_{i_k}|+|u|}(x_{i_1} \dots x_{i_k} u \oplus \widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k}) \\ &= \mathit{Pref}_{|x_{i_1} \dots x_{i_k}|+|u|}(x_{i_1} \dots x_{i_k} u x_L \oplus \widetilde{x}_L \widetilde{x}_{i_1} \dots \widetilde{x}_{i_k} u') \end{aligned}$$

with $u' = \widetilde{x}_{i_{k+1}} \dots \widetilde{x}_{i_{k+d}}$. We obtain $z = \mathit{Pref}_{|z|}(\beta(s))$ with $s = b_{i_1} \dots b_{i_{k+d}}$.

The reverse direction follows by remarking that every previous statement is invertible. □

4. The Graph Associated to the Transducer $\mathcal{T}(X)_{f,x_L}$

In this section we examine the transducer $\mathcal{T}(X)_{f,x_L} = \langle Q, i, \delta, F \rangle$, defined in the previous sections, from the point view of graph theory. According to the definition of Latin square, given two transitions $\delta((u, v), c) = (b_j, (u', v'))$ and $\delta((u, v), d) = (b_k, (u", v"))$ in $\mathcal{T}(X)_{f,x_L}$, if $c \neq d$, then we have $(u', v') \neq (u", v")$. This leads to introduce a graph, namely $G(X)_{f,x_L}$, which is obtained by removing the labels from the transitions of $\mathcal{T}(X)_{f,x_L}$. More precisely:

- We associate to every transition $\delta((u, av), c) = (\varepsilon, (ub, v))$ in $\mathcal{T}(X)_{f,x_L}$, the edge $(u, av) \to (ub, v)$ in $G(X)_{f,x_L}$.
- We associate to every transition $\delta((u, av), c) = (b_{i_1}, (x_{i_2} \dots x_{i_{d+1}}, v\widetilde{x}_{i_1}))$, the edge $(u, av) \to (x_{i_2} \dots x_{i_{d+1}}, v\widetilde{x}_{i_1})$ in $G(X)_{f,x_L}$.

A natural question arises: what happens to $G(X)$ when the Latin square map is replaced by another one?

4.1. *Exchanging the Latin square map*

Proposition 15. *Given a vertex* (u, av) *of* $G(X)_{f,x_L}$ *and given a letter* $b \in A$, *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* $G(X)_{f,x_L}$.
(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* $G(X)_{f,x_L}$.

As a direct consequence, we obtain the following property:

Corollary 16. *Given two Latin squares maps f, g, the corresponding graphs $G(X)_{f,x_L}$ and $G(X)_{g,x_L}$ are isomorphic.*

Actually, this result shows that any graph $G(X)_{f,x_L}$ is actually independent of the Latin square f: this fact corroborates the validity of the generic notation $\oplus$ for any Latin square, and it allows to denote by $G(X)_{x_L}$ the graph $G(X)_{f,x_L}$.

Moreover, we remark another property: the transducer $\widetilde{\mathcal{T}}(X)_{f,x_L}$ can be realized by a transducer $\mathcal{T}(X)_{f,x_L}$. Indeed, each state (u, v) of $\mathcal{T}(X)_{f,x_L}$ can be associated to the state $(\widetilde{u}, \widetilde{v})$ of $\widetilde{\mathcal{T}}(X)_{f,x_L}$, the transition being in one to one correspondence as indicated in the following: With each transition $\widetilde{\delta}((\widetilde{u}, \widetilde{v}), c) = (b, (\widetilde{u}', \widetilde{v}'))$ which is defined in $\widetilde{T}(X)_{f,x_L}$, with $b \in B \cup \{\varepsilon\}$ we associate the unique transition $\delta((u, v), c) = ((u', v'), b)$ in $\widetilde{T}(X)_{f,x_L}$.

Corollary 17. *Let $\mathcal{T}(X)_{f,x_L}, \widetilde{\mathcal{T}}(X)_{f,x_L}$ be the transducers defined as indicated above. Then the corresponding graphs $G(X)_{x_L}$ and $\widetilde{G}(X)_{x_L}$ are isomorphic. Moreover if f is a commutative Latin square then they are also isomorphic as transducers.*

Remark 18. *This result is consistent with the one in* [4]*, where the isomorphism between the left-to-right and right-to-left decoding transducers is proved for binary codes, where the only Latin square map, different from identity, that is also commutative, is the bitwise sum.*

4.2. *A remarkable strongly connected component*

Another natural question is to ask what happens to the graph $G(X)_{x_L}$ when the encoding key x_L is modified.

In Example 8, we note that the unlabelled graphs associated to the two corresponding transducers contains only one non trivial strongly connected component. Actually, this illustrates a general property of transducers $\mathcal{T}(X)_{x_L}$ that we are going to show in the sequel.

We consider $C(G(X)_{x_L})$, the subgraph of $G(X)_{x_L}$ whose vertices are elements of $V = X^d(\mathit{Pref}(X) \setminus X) \times \mathit{Suff}(\widetilde{X}^+)$ and whose edges are the edges in $G(X)_{x_L}$ which are connected to V. connecting vertices in the subgraph.

Proposition 19. *$C(G(X)_{x_L})$ is a strongly connected component of $G(X)_{x_L}$.*

In view of proving Proposition 19, we state the following result that comes from construction:

Lemma 20. *Let $(x_{i_1} \dots x_{i_d} p, v)$ be a vertex of $C(G(X)_{x_L})$, with $x_{i_j} \in X$, for all $1 \le j \le d$, and let $x \in X$ such that p is a prefix of x. Then $C(G(X)_{x_L})$ contains a path from $(x_{i_1} \dots x_{i_d} p, v)$ to $(x_{i_2} \dots x_{i_d} x, r\widetilde{x}_{i_1})$, where r is the suffix of v with length $|x| - |p|$.*

Proof of Proposition 19. We consider an arbitrary pair of vertices in $C(G(X)_{x_L})$, $(t_1 \dots t_d p, v), (t'_1 \dots t'_d q, v')$, with $t_i, t'_j \in X$ for all $1 \leq i, j \leq d$; $p, q \in \mathit{Pref}(X) \setminus X$ and $v, v' \in \mathit{Suff}(\widetilde{X}^+)$. We are going to prove that $C(G(X)_{x_L})$ contains a path from $(t_1 \dots t_d p, v)$ to $(t'_1 \dots t'_d q, v')$. More precisely, we are going to construct a sequence of vertices $(u_i, v_i)_{i=1}^m$ of $C(G(X)_{x_L})$ such that $(u_0, v_0) = (t_1 \dots t_d p, v)$, $(u_m, v_m) = (t'_1 \dots t'_d q, v')$ and such that $C(G(X)_{x_L})$ contains a path from (u_k, v_k) to (u_{k+1}, v_{k+1}) for all $0 \leq k \leq m-1$. By definition, there exists a pair of words $t_{d+1} \in X$, $y \in \widetilde{X}^+$ such that p is a proper prefix of t_{d+1} and such that v' is a suffix of y. Without loss of generality, we may assume that $y = \widetilde{x}^{d+k}$ for some $k \geq 1$ and $x \in X$.

Starting from $u_0 = t_1 \dots t_d p, w_0 = v_0 = v$, we set $u_1 = t_2 \dots t_{d+1}$, $w_1 = w_0 \widetilde{t_1} = v\widetilde{t_1}$, and we denote by v_1 the suffix of w_1 of lenght $L - |u_1|$. According to Lemma 20, there is in $C(G(X)_{x_L})$ a path from (u_0, v_0) to (u_1, v_1).

Since u_1 starts with t_2 we consider $u_2 = (t_2)^{-1} u_1 x = t_3 \dots t_{d+1} x$, $w_2 = w_1 \widetilde{t_2} = v\widetilde{t_1}\widetilde{t_2}$, and denote by v_2 the suffix of w_2 with lenght $L - |u_2|$. Furthermore according to Lemma 20, a path from (u_1, v_1) to (u_2, v_2) exists in $C(G(X)_{x_L})$.

The construction is completed in a similar way. The main steps are summarized as indicated below. In each step, v_n stands for the suffix of w_n with length $L - |u_n|$.

$$
\begin{array}{ll}
u_0 = t_1 \dots t_d p, & w_0 = v_0 = v \\
u_1 = t_2 \dots t_{d+1}, & w_1 = w_0 \widetilde{t_1} = v\widetilde{t_1} \\
u_2 = t_3 \dots t_{d+1} x, & w_2 = w_1 \widetilde{t_2} = v\widetilde{t_1}\widetilde{t_2} \\
\vdots & \vdots \\
u_d = t_{d+1} x^{d-1}, & w_d = v\widetilde{t_1} \dots \widetilde{t_d} \\
u_{d+1} = x^d, & w_{d+1} = v\widetilde{t_1} \dots \widetilde{t_d}\widetilde{t_{d+1}} \\
u_{d+2} = x^d, & w_{d+2} = v\widetilde{t_1} \dots \widetilde{t_{d+1}}\widetilde{x} = w_{d+1}\widetilde{x} \\
\vdots & \vdots \\
u_{d+k-1} = x^d, & w_{d+k-1} = w_{d+1}\widetilde{x}^{k-1} \\
u_{d+k} = x^{d-1} t'_1, & w_{d+k} = w_{d+1}\widetilde{x}^k \\
\vdots & \vdots \\
u_{2d+k-1} = x t'_1 \dots t'_{d-1}, & w_{2d+k} = w_{d+1}\widetilde{x}^{d+k-1} \\
u_{2d+k} = t'_1 \dots t'_d, & w_{2d+k+1} = w_{d+1}\widetilde{x}^{d+k} \\
u_{2d+k+1} = t'_1 \dots t'_d q, & w_{2d+k+2} = w_{2d+k+1}\,.
\end{array}
\tag{1}
$$

According to Lemma 20, for each integer $i \geq 0$, a path from (u_i, v_i) to (u_{i+1}, v_{i+1}) exists in $C(G(X)_{x_L})$. This completes the proof of Proposition 19. As a consequence, we state the following result:

Proposition 21. *Given a code X with a finite deciphering delay, and given a encoding 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 $G(X)_{x_L}$.*

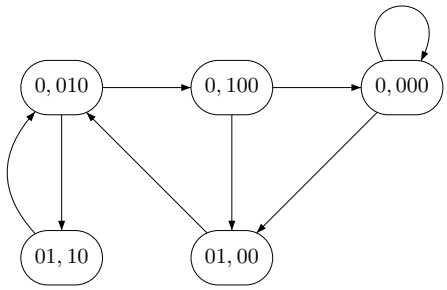

Fig. 4. Theorem 22: the connected graph $C(G(X)$ associated to $X = \{0, 01\}$.

Proof. Let X be a finite code with finite deciphering delay $d \geq 0$. First, we prove that the strongly connected component $C(G(X)_{x_L})$ is accessible from any vertex of $G(X)_{x_L}$. Let (u, v) be a vertex in $G(X)_{x_L} \setminus C(G(X)_{x_L})$. We have $v = v_1 v_2$, with v_1 a non empty suffix of $\widetilde{x}_L$ and $v_2 \in \mathit{Suff}(\widetilde{X}^+)$. Moreover, we have $u = u'p$ with $u' \in X^+$ and $p \in \mathit{Pref}(X)$. By construction, a path with length not smaller than $|v_1|$ starting from (u, v) necessarily reach a vertex (u', v'), with $v' \in \mathit{Suff}(\widetilde{X}^+)$, thus (u', v') is a vertex in $C(G(X)_{x_L})$. This proves that $C(G(X)_{x_L})$ is accessible from any vertex of $G(X)_{x_L}$.

In order to prove the uniqueness of $C(G(X)_{x_L})$ we are going to prove that, for any (u, v) in $G(X)_{x_L} \setminus C(G(X)_{x_L})$, the strongly connected component of (u, v) is $\{(u, v), \emptyset\}$. In fact, let (u', v') be a vertex in $G(X)_{x_L} \setminus C(G(X)_{x_L})$ such that there is a path from (u, v) to (u', v'). By construction, a unique pair of words v_1, v_2 and a unique pair of words v'_1, v'_2 exist such that $v = v_1 v_2$ and $v' = v'_1 v'_2$, with v_1,v'_1 non empty suffixes of $\widetilde{x}_L$, v_2, $v'_2 \in \mathit{Suff}(\widetilde{X}^+)$, and $|v'_1| < |v_1|$. This last condition implies that $(u', v') \neq (u, v)$ and so that no cycle of $G(X)_{x_L}$ may contain (u, v), and this completes the proof. □

As another remarkable property, following by construction and by Proposition 21, the component $C(G(X)_{x_L})$ does not depend of the key x_L.

Theorem 22. *Given a code X with finite deciphering delay d, a unique strongly connected graph, namely $C(X)$, exists such that $C(X) = C(G(X)_{x_L})$, for any key x_L.*

Figure 4 illustrates the result of Theorem 22 with the condition of Example 8.

5. Conclusions

In this paper we gave a generalization of Girod's encoding/decoding method to codes defined on alphabets of any size and with a finite deciphering delay. We also gave a general construction of a transducer for decoding texts encoded by the generalized Girod's method. Some properties of this transducer and its underlying

graph have been investigated. As a further works it could be of interest to develop similar methods not just for finite codes but also for rational codes.

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, 2010.

[2] J. Berstel, D. Perrin and C. Reutenauer. *Codes and Automata*. Cambridge University Press, 2010.

[3] A. S. Fraenkel and S. T. Klein. Bidirectional Huffman Coding, *The Computer Journal*, 33:296307 (1990)

[4] L. Giambruno and S. Mantaci. Transducers for the bidirectional decoding of prefix codes, *Theoretical Computer Science*, 411:1785-1792 (2010)

[5] 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)

[6] B. Girod. Bidirectionally decodable streams of prefix code words. *IEEE Communications Letters*, 3(8):245–247, August 1999.

[7] M. Lothaire. *Applied combinatorics on words*, Vol 104 of Encyclopedia of mathematics and its applications. Cambridge University Press, 2005.

[8] J. Sakarovitch. *Éléments de théorie des automates*. Vuibert Informatique, 2003.

[9] D. Salomon. *Variable-Length Codes for Data Compression*. Springer-Verlag, 2007.