**Theoretical Informatics and Applications** Will be set by the publisher
Informatique Théorique et Applications

# ON THE SIZE OF TRANSDUCERS FOR BIDIRECTIONAL DECODING OF PREFIX CODES

Laura Giambruno[1] and Sabrina Mantaci[1]

**Abstract**. In a previous paper [5] a bi-deterministic transducer is defined for the bidirectional decyphering of words by the method introduced by Girod [6]. Such a method is defined using prefix codes. Moreover a coding 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. It is proved also that this transducer is minimal.

Here we consider the number of states of such a transducer, related to some features of the considered prefix code $X$. We find some bounds of such number of states in relation with different notions of "size" of $X$. In particular, we give an exact formula for the number of states of transducers associated to maximal prefix codes, and we consider two special cases of maximal uniform codes and a class of codes, that we named string-codes, showing that they represent, for maximal codes, the extremal cases to this number of states in terms of different sizes. This fact formalizes somehow the intuition that for maximal prefix codes the farthest a code is from being uniform the greatest the number of states is in the correspondent transducer, respect to the different sizes of the code.

Moreover we prove that prefix codes corresponding to isomorphic trees have transducers that are isomorphic as unlabeled graphs.

**1991 Mathematics Subject Classification.** 94B35, 68P30, 94A45.

## 1. Introduction

There are many reasons for decoding a message in both directions. The most important is connected to data integrity. In fact when we use a variable length

*Keywords and phrases:* variable length codes, prefix codes, bilateral decoding, transducers

[1] Dipartimento di Matematica e Applicazioni, Università di Palermo, Via Archirafi 34, 90123 Palermo, Italy ; e-mail: lgiambr@math.unipa.it & sabrina@math.unipa.it

code (VLC in short) for source compression (cf. [1], [8]), 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. If a single error occurs in a record, the decoder tries to read the record from the end to the beginning. If there is just one bit error in the coding of the record, it is possible to avoid the error propagation and isolate it. In order to do this we need codes that can be easily decoded in both directions. These are called *reversible variable length codes* (RVLCs in short). Actually RVLCs are usually big and difficult to be constructed (cf. [4]), whereas prefix codes over a $k$-letter alphabet, i.e. sets of words where no word is a prefix of another one, are very easy to be found, since they are in bijection with $k$-ary trees. A word encoded by a prefix code can be easily decoded from left to right without any delay, but it looses this property when we try to decode it from right to left.

Moreover, a very strong result for maximal prefix code due to Schutzenberger (cf. [3]), states that a maximal finite code is either prefix or with an infinite deciphering delay. This means that for maximal codes the only ones that can be decoded in both directions with a finite deciphering delay are the bifix codes, i.e. the codes that are both prefix and suffix.

In 1999 Girod (cf. [6]) introduced a very interesting alternative method to encode words by using prefix binary codes, that allows to decode the encoded word both from left to right and from right to left with a literal deciphering delay of at most the length of the longest word in the code.

In [5] we introduced a construction for a transducer that allows the bidirectional decoding of messages encoded by Girod's method. We also introduced a variant of the Girod's coding method, and we defined a transducer that allows both right-to-left and left-to-right decoding. We also proved that this transducer is deterministic, co-deterministic and minimal.

For sake of completeness, in this paper we recall Girod's encoding method with its variant and the construction of the transducer associated to the decoding operation on a given code $X$. Here we are mainly interested to find some bounds to the number of states of this transducer, depending on different notions of "size" of the prefix code $X$, such as the cardinality of $X$, the length of $X$, i.e. the sum of the lengths of its words, the number of nodes of the tree representing $X$, and the length of the longest word in $X$.

In Section 2 we introduce some preliminary definitions and properties regarding codes and transducers, and the connection between these two notions. We moreover describe the method introduced by Girod for the bidirectional decoding of a prefix word and its variant (see [5]). We describe the construction of the transducer associated to its variant and we recall some of its properties.

In Section 3 we prove some general results on the number of states of the transducer associated to prefix codes. In particular we give a general upper bound that holds for transducers associated to any prefix code. We prove that for prefix codes

corresponding to isomorphic trees, the corresponding transducers are isomorphic as unlabeled graphs. In Section 4 we focus in particular our attention to maximal prefix codes, for which we find a precise formula giving the number of states of the associated transducer. This allows to prove, for maximal codes, an exponential lower bound on the length of the longest codeword. The formula gives, as particular cases, the one of uniform maximal codes and one of the so called string-codes. In particular, transducers associated to maximal uniform trees have a number of states that is linear with the length of the code, and loglinear in the cardinality of the code and in the size of the tree associated to the code, whereas the size of the transducer associated to string-codes is exponential in all these notions of size. Finally, in Section 5, we give the number of states of the transducer associated to a uniform (non maximal) code with two words, and an upper bound in the general case of non maximal prefix code.

In Section 6 we give some conclusions and open problems.

## 2. Preliminaries

### 2.1. 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 on $A$. 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 $\gamma$ 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 $\gamma$. 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 $Y$.

A set over $A^*$ is called a *bifix set* if it is both a prefix and a suffix set. It can be easily proved that prefix, suffix and bifix sets are codes, called respectively *prefix codes*, *suffix codes* and *bifix codes*. A code is *maximal* if it is not contained in any other code. A prefix code is maximal if and only if it is maximal as a prefix code. A *decoding* is the inverse operation than encoding i.e. the decoding of $\gamma$ is the function $\gamma^{-1}$ restricted to $\gamma(B^*)$.

We say that a set $X \subset A^+$ is *weakly prefix*, or that *has a literal deciphering delay* $d$, if there exists an integer $d \geq 0$ such that if $xu$ is a prefix of $x'y'$ with $x, x' \in X$, $u$ a prefix of a word in $X^*$, and $y' \in X^*$, then $|u| < d$ implies $x = x'$.

Throughout this paper we consider codes over a binary alphabet, that is $A = \{0, 1\}$. For each word $u$ we denote by $\tilde{u}$ the reverse of $u$. For $X = \{x_1, x_2, \dots, x_n\}$, we define by $\widetilde{X}$ the set $\widetilde{X} = \{\widetilde{x}_1, \widetilde{x}_2, \dots, \widetilde{x}_n\}$.

A finite transducer $T$ uses an input alphabet $A$ and an output alphabet $B$. It consists of a quadruple $T = (Q, I, F, E)$ where $Q$ is a finite set whose elements are called *states*, $I$ and $F$ 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 with first element the input alphabet and second element the output alphabet. A transducer $T$ 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. This is called the *relation realized by* $T$. A transducer is called a *literal transducer* if each input label is a single letter. A literal transducer is called *deterministic* (resp. *codeterministic*) if for each state $p$ and for each input letter $a$ there is at most one edge starting at (resp. ending at) $p$ with input letter $a$.

We can represent encoding and decoding using transducers. An encoding $\gamma$ 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. An interesting result is that for any encoding there exists a literal unambiguous transducer which realizes the associated decoding (see [**?**, 7]).

A *sequential transducer* over $A$, $B$ is a triple $T = (Q, i, F)$ together with 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 $Q \times A \longrightarrow B^*$. In addition, the initial state $i \in Q$ has attached a word $\lambda$ called the *initial prefix* and $F$ is partial function $F : Q \longrightarrow B^*$ called the *terminal function*. Thus, an additional prefix and additional suffix can be attached to all the outputs. By definition, a sequential transducer is deterministic. There is a unique *minimal sequential transducer* equivalent to a given one i.e. with the minimal number of states among the sequential transducers realizing the same relation (cf. [7]).

### 2.2. Girod's method and transducers

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 particular, in the case of maximal prefix codes, we have a very strong result due to Schutzenberger (cf. [3]) that states the following:

**Theorem 2.1** (**Schutzenberger**). *A finite maximal code with finite deciphering delay is prefix.*

This means that the only maximal codes that can be bidirectionally decoded with finite delay are the ones that are both prefix and suffix.

In 1999 B. Girod (cf. [6]) introduced a very interesting alternative coding method using finite prefix codes, that besides the simple concatenation of codewords, applies a transformation to the obtained word that allows the deciphering of the coded word in both directions, with a literal deciphering delay bounded by the length of the longest word in the code. This is somehow surprising, since, by

paying just a small deciphering delay from right to left, we get a finite deciphering delay from left to right, even if the code is just prefix, but not necessarily suffix.

In this section we describe this coding method, due to Girod, where, given a finite prefix code $X$, any sequence of codewords in $X$ is transformed in a bitstring that can be decoded in both directions, with a literal delay of the length of the longest string in the code.

Such a method is based on a well-known property of the binary sum. The binary sum operation $\oplus$ is a binary operation on $\{0,1\}$ that returns a bit in this way: for $a, b$ either both 1 or both 0, $a \oplus b$ returns 0 and in the other cases it returns 1. For the operation $\oplus$ the following property holds: if $c = a \oplus b$ then $b = a \oplus c$ and $a = b \oplus c$.

Let $X = \{x_1, \ldots, x_m\}$ be a finite prefix code defined by an encoding $\gamma$ over an alphabet $B = \{b_1, \ldots, b_m\}$. Consider a word $w = b_{i_1} \ldots b_{i_k}$ in $B^*$ and its encoding $y = \gamma(b) = x_{i_1} \ldots x_{i_k}$ where $x_{i_j}$'s are codewords in $X$. By concatenating the reverse of each codeword $x_{i_j}$, we obtain the word $y' = \tilde{x}_{i_1} \ldots \tilde{x}_{i_k}$. Let $z = y \oplus y'$. The idea would be to decode $y$ from $z$ using the relation $y = z \oplus y'$. Anyway we cannot apply this idea since we should know $y'$ in order to decode $y$. However we know that the elements in $y'$ are strictly related to those in $y$. If we lightly modify $y$ and $y'$ we obtain the solution given by Girod.

Let us denote by $L$ the length of the longest codeword in $\{x_{i_1}, \ldots, x_{i_k}\}$ and let us append the word $0^L$ to $y$ as a suffix and to $y'$ as a prefix. Then consider the words $x = y0^L$, $x' = 0^L y'$ and $z = x \oplus x'$ and define the encoding $\delta$ from $B^*$ to $A^*$ such that, for any $w = b_{i_1} \cdots b_{i_k} \in B^*$, $\delta(w) = z$, where $z$ is defined as before.

Since the first $L$ bits of $x'$ are 0's, then the first $L$ bits of $z$ are equal to the first $L$ bits of $x$. By the definition of $L$, those $L$ bits contain as prefix at least the first codeword $x_{i_1}$ in $y$. We concatenate its reverse $\tilde{x}_{i_1}$ to $x'$. In this way $x'$ has again $L$ unread symbols, that can be summed to the next $L$ symbols of $z$. As before, this sum contains as prefix at least the second codeword $x_{i_2}$. Its reverse can be again concatenated to $x'$ and have again $L$ unread bits in $x'$. By proceeding in this way we obtain the left-to-right decoding of $z$.

Similarly we can decode $z$ from right to left: in this case we invert the roles of $x$ and $x'$ and apply the operation $\oplus$ to $z$ and to bits of $x$ from right to left in order to obtain new bits for $x'$.

In [5], we remark that by using the properties of $\oplus$ the method can be analogously applied when any word of length $L$ is used in the place of $0^L$. We choose to use among the words of maximal length in $X$, the one that is minimal in lexicographic order (given a code, it is univocally determined). We refer to it as the *Girod's generalized method*.

We describe (see [5]) how to construct a transducer for the generalization of Girod's left-to-right decoding. For the description of the transducer for the classic Girod's left-to-right decoding see [5].

Let $X = \{x_1, \ldots, x_m\}$ be a finite prefix code defined by an encoding $\gamma$ over an alphabet $B = \{b_1, \ldots, b_m\}$. Let $L$ be the length of the longest word in $X$ and let $x_L$ be the smallest word in the lexicographic order among the words in $X$ of length

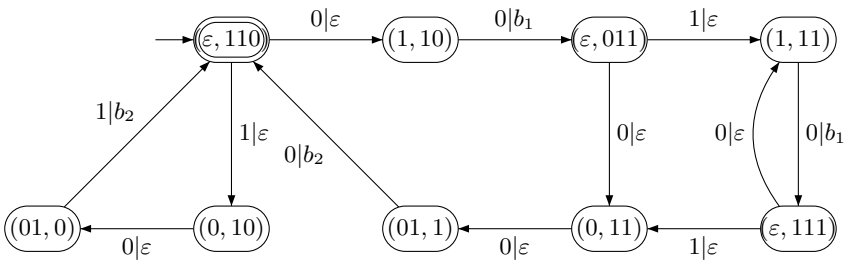


FIGURE 1. The transducer $T$ for the left-to-right decoding by the Girod's generalized method of $X = \{11, 011\}$ over $B = \{b_1, b_2\}$

$L$. For any sequence $y$ of codewords in $X$ we consider the encoding $\delta_{x_L}$ as defined by the generalization of Girod's method. In order to simplify the notation we use $\delta_L$ instead of $\delta_{x_L}$. The transducer $T = (Q, i, F, E)$ for the left-to-right decoding of $\delta_L$ is defined as follows.

The states in $Q$ are pairs of words $(u, v)$ such that:

- $u$ is a proper prefix of a word in $X$;
- $v$ is a suffix of a word in $\tilde{x}_L \widetilde{X}^*$ of length $L - |u|$;

The unique initial and final state $i$ is $(\epsilon, \tilde{x}_L)$.

The edges in $E$ are defined as follows:

(1) $((u, av), c, \epsilon, (ud, v))$, with $a \oplus c = d$, if $ud \notin X$ and $ud$ is a prefix of a word in $X$
(2) $((u, av), c, b_i, (\epsilon, vd\tilde{u}))$, with $a \oplus c = d$, if $ud = x_i \in X$

In all remaining cases the transitions are not defined.

In Figure 1 we show the transducer $T$ for the decoding of $X = \{11, 011\}$.
In [5] it is proved the following:

**Theorem 2.2.** *The transducer $\mathcal{T}$ realizes the function $\varphi$ defined by $\varphi(z) = \delta_L^{-1}(z)b_L$, where $\delta_L^{-1}$ is the decoding of $\delta_L$ from left to right and $b_L$ is the word $\gamma^{-1}(x_L)$. Moreover this transducer is deterministic, co-deterministic and minimal.*

In a similar way we can define a transducer for the right to left encoding. In [5] we prove that:

**Theorem 2.3.** *The transducers for the left-to-right and for the right-to-left decoding are isomorphic.*

This means that we can use the same transducer for decoding a word in both directions.

## 3. General bounds and properties

In the present and in the next two sections we are interested in counting the number of states of the transducer associated to a given prefix code $X$, depending

on some of its features. We consider the dependence of this number on the different notions of "size" of $X$, such as the number of codewords in $X$, the sum of their lengths, the length of the longest word, and the dimension of the tree representing $X$. In this section we consider some general bounds and we prove that codes associated to isomorphic trees have transducers that are isomorphic themselves as unlabeled graphs.

Given a prefix code $X = \{x_1, x_2, \ldots, x_m\}$ we can measure the *size* of $X$ in different ways:

- $|X|$, the *cardinality* of $X$, that is, the number of its elements;
- $\|X\| = \sum_{x \in X} |x|$, the *length* of $X$, i.e. the sum of the lengths of its words;
- $|T_X|$, the size of the binary tree $T_X$ naturally associated to $X$, i.e. the number of its nodes;
- $L = \max_{x \in X} |x|$, the length of the longest word in $X$.

For $X$ prefix code, let $\mathcal{T}_X = (Q, i, F, E)$ be the transducer for its bidirectional decoding by the Girod's generalized method, with $(\varepsilon, \widetilde{x}_L)$ as initial state. We are interested to find a bound to $|Q|$.

The following theorem gives a general upper bound for the number of states of the transducer associated to any prefix code.

**Theorem 3.1.** *If $X$ is a prefix code then $|Q| \leq L\ 2^L$.*

*Proof.* Every state is a pair of words $(u, v)$ such that $|u| + |v| = L$. The number of all possible pairs with this constraint is exactly $L\ 2^L$. In fact this corresponds to consider any of the $2^L$ words $w$ of length $L$, and to factorize it as the concatenation of two words, in all possible ways. There exist exactly $L$ of such factorizations. This concludes the proof.

□

The following lemma shows that the number of states of the transducer grows when adding words to the prefix code:

**Lemma 3.2.** *Let $Y \subseteq X$ be prefix codes such that the length of the longest word in $X$ is equal to length of the longest word in $Y$. Then $\mathcal{T}_Y$ is contained in $\mathcal{T}_X$ and the number of states of $\mathcal{T}_Y$ is strictly less than the number of states of $\mathcal{T}_X$.*

*Proof.* In order to prove the result, it is sufficient to prove that the transducer for a prefix code $Y = X \cup \{x\}$ is obtained from $\mathcal{T}_X$ by adding states and transitions. This fact follows by construction. In fact, consider the transducer $\mathcal{T}_X$. Any path from a state $(\epsilon, v\tilde{x}_i)$ to the state $(\epsilon, v'\tilde{x}_j)$, where $v, v'$ are suffixes of $\tilde{X}$ allows to decode $x_j$ after the decoding of $x_i$. All the states in these paths should be preserved when the new word is added, so no state is ever deleted. The introduction of the new word cause the addition of new states and transitions. In fact at least a state of acceptance for $x$, $(\epsilon, u\tilde{x})$ must be added. □

Given two binary trees $T_1$ and $T_2$, we say that they are isomorphic if $T_2$ can be obtained from $T_1$ by choosing some of its nodes and, for each of them, switching

the right and the left subtree. We say that two *prefix codes* are *isomorphic* if the associated trees are isomorphic. We have the following theorem:

**Theorem 3.3.** *If $X$ and $Y$ are two isomorphic prefix codes then the corresponding transducers are isomorphic as unlabeled graphs.*

In order to prove it we first prove the following:

**Proposition 3.4.** *If $X$ and $Y$ are two isomorphic prefix codes such that $T_X$ is obtained by $T_Y$ by a single rotation around a given vertex, then the corresponding transducers are isomorphic as unlabeled graphs.*

*Proof.* Let $X$ and $Y$ be two isomorphic codes, such that $T_X$ is obtained by $T_Y$ by a single rotation around a given vertex $V$. We denote by $z$ the word corresponding to the path in $T_X$ from the root to $V$.

We can decompose $X$ as $X_1 \cup X_2$, where $X_1$ is the set of words in $X$ that have $z$ as a prefix and $X_2$ the set of all the remaining words in $X$. Let $X_1 = \{za_1y_1, \ldots, za_ky_k\}$. If $a$ is a bit, we denote by $\bar{a}$ its opposite. By the definition of rotation, it follows that $Y = Y_1 \cup X_2$, with $Y_1 = \{z\bar{a}_1y_1, \ldots, z\bar{a}_ky_k\}$.

Let us consider the function $\varphi$ between the set of prefixes of words in $X$ and the set of prefixes of the words in $Y$,

$$\varphi : Pref(X) \longrightarrow Pref(Y)$$

such that, for each $u$ in $Pref(X)$,

- $\varphi(u) = z\bar{a_i}w$, if $u = za_iw$ for some bit $a_i$ and some word $w$
- $\varphi(u) = u$ otherwise.

Let $Q_X$ be the set of states of $\mathcal{T}_X$ and $Q_Y$ be the set of states of $\mathcal{T}_Y$. We define the map $\overline{\varphi} : Q_X \longrightarrow Q_Y$ in the following way.
For a state $(u, v) \in Q_X$ with $v = \tilde{s}\tilde{y_1} \ldots \tilde{y_r}$ *and* $s \in Pref(X)$, $\{y_1 \ldots y_r\} \subseteq X$, we define

$$\overline{\varphi}((u, v)) = (\varphi(u), \widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \ldots \widetilde{\varphi(y_r)})$$

The function $\varphi$ is well defined since if $u$ is a prefix of a word in $X$ then $\varphi(u)$ is a prefix of a word in $Y$, and if $v$ is a suffix of a word in $\widetilde{X}^*$ of length $L - |u|$ then $\widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \ldots \widetilde{\varphi(y_r)}$ is a suffix of a word in $\widetilde{Y}^*$ of length $L - |\varphi(u)|$. Moreover is easy to see that by construction $\varphi$ is injective and surjective.

In order to be an isomorphism of unlabeled graphs we have to prove that $\overline{\varphi}$ preserves the edges. In particular, we have to prove that if $((u, v), \ell, y, (u', v'))$ is an edge in $\mathcal{T}_X$ then $(\overline{\varphi}((u, v)), \ell', y, \overline{\varphi}((u', v')))$ is an edge in $\mathcal{T}_Y$.

Let us consider a state $(u, av)$ in $Q_X$, with $av = \widetilde{s}\widetilde{y_1} \ldots \widetilde{y_r}$, $s \in Pref(X)$ and $y_1 \ldots y_r$ words in $X$. Let us consider a letter $\ell$ such that if $b = a \oplus \ell$, $ub$ is either a prefix in $X$ or a word in $X$.

We consider separately the two different cases.

(1) Consider $(u, av)$ such that $ub$ is a proper prefix of a word in $X$. Thus we have in $\mathcal{T}_X$ the transition $((ua, v), \ell, \varepsilon, (ub, v))$. We have that $\overline{\varphi}(u, av) = (\varphi(u), \widetilde{\varphi(sa)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)})$. Let $c$ be the letter such that $\widetilde{\varphi(sa)} = c\widetilde{\varphi(s)}$. Note that $c$ is equal to $\bar{a}$ or to $a$ depending on $s = z$ or not.

On the other side we have that $\overline{\varphi}(ub, v) = (\varphi(ub), \widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)})$. Let $d$ be the letter such that $\varphi(ub) = \varphi(u)d$. Here $d$ is equal to $\bar{b}$ or to $b$ depending $u = z$ or not.

We conclude the proof of case 1., since there must be in $\mathcal{T}_Y$ the transition from $(\varphi(u), c\widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)})$ to $\overline{\varphi}(ub, v) = (\varphi(u)d, \widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)})$ with label $(c \oplus d, \varepsilon)$.

(2) Let $(u, av)$ be such that $ub$ is the word $x_i$ in $X$. Thus we have in $\mathcal{T}_X$ the transition $((ua, v), \ell, b_i, (\varepsilon, \widetilde{vub}))$.

We have that $\overline{\varphi}(u, av) = (\varphi(u), \widetilde{\varphi(sa)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)})$. Let $c$ be the letter such that $\widetilde{\varphi(sa)} = c\widetilde{\varphi(s)}$. $c$ is equal to $\bar{a}$ or to $a$ depending on $s = z$ or not.

On the other side we have that $\overline{\varphi}(\varepsilon, \widetilde{vub}) = (\varepsilon, \widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)}\widetilde{\varphi(ub)})$. Let $d$ be the letter such that $\varphi(ub) = \varphi(u)d$. The value of $d$ will be equal to $\bar{b}$ or to $b$ depending on $u = z$ or not.

We conclude the proof in this case since there must be in $\mathcal{T}_Y$ the transition from $(\varphi(u), c\widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)})$ to $\overline{\varphi}(\varepsilon, \widetilde{vub}) = (\varepsilon, \widetilde{\varphi(s)}\widetilde{\varphi(y_1)} \dots \widetilde{\varphi(y_r)}\widetilde{\varphi(ub)})$ with label $(c \oplus d, \varepsilon)$.

□

*Proof.* (Theorem 3.3) If $T_X$ and $T_Y$ are two isomorphic trees, then $T_Y$ can be obtained from $T_X$ by a finite number of switches of subtrees around some given vertices. Induction, combined with Proposition 3.4 concludes the proof.

□

## 4. Bounds for maximal prefix codes

In this section we consider maximal prefix codes. It is well known that maximal prefix codes are represented by complete trees, i. e. trees where each node has two or zero subtrees. For maximal prefix codes we prove an exact formula that computes the number of states of the associated transducer. As a consequence, we get an exponential lower bound on $L$ for this number. It is well known (see [2]) that, if $X$ is a maximal prefix code, any word $w$ in $A^*$ is also in $X^*Pref(X)$ that is, it can be written as concatenation of a sequence of codewords followed by a prefix of a codeword. Consequently $\tilde{w}$ is in $Suff(\tilde{X})\tilde{X}^*$.

**Lemma 4.1.** *If $X$ is a maximal prefix code then, for each pair of words $(u, v)$ with $u = \epsilon$ or $u$ proper prefix of $X$ and $v \in A^{L-|u|}$, we have that $(u, v) \in Q$.*

*Proof.* Let $w \in A^L$. Let us prove that $(\varepsilon, \tilde{w})$ is accessible.

By construction, $\tilde{w} \in Suff(\tilde{X})\tilde{X}^*$, then $\tilde{w}$ can be written as $\tilde{w} = \tilde{x}'_0\tilde{x}_1...\tilde{x}_t$ with $\tilde{x}'_0$ suffix of some $\tilde{x}_0 \in \tilde{X}$. Let $w_0 = \tilde{x}_0...\tilde{x}_t$, and consider the sum $z = x_0...x_t \oplus pref_{-L}(\tilde{x}_L\tilde{x}_0...\tilde{x}_t)$. By reading label $z$ starting from the initial state $(\varepsilon, \tilde{x}_L)$ we get to the state $(\varepsilon, \tilde{w})$. In fact in [5] we have proved that there exists a path from the initial state to a state $(\epsilon, u)$ with label $(z, b_{i_1} \dots b_{i_k})$ if and only if $z = x_{i_1} \dots x_{i_k} \oplus pref_{-L}(\tilde{x}_L\tilde{x}_{i_1} \dots \tilde{x}_{i_k})$. Thus, given $z = x_0...x_t \oplus pref_{-L}(\tilde{x_L}\tilde{x_0}...\tilde{x_t})$, we have a path from the initial state to a state of the form $(\varepsilon, t)$ with label $(z, b_0 \dots b_t)$. Since in [5] we have proved that if there exists in $T$ a path from the initial state $(\epsilon, \tilde{x}_L)$ to a state of the form $(\epsilon, u)$, with label $(z, b_{i_1} \dots b_{i_k})$, then $u$ is the suffix of length $L$ of $\tilde{x}_L\tilde{x}_{i_1} \dots \tilde{x}_{i_k}$, we have that the terminal state of this path is $(\varepsilon, \tilde{w})$. Thus the state $(\varepsilon, \tilde{w})$ is accessible.

Notice that, by construction, from every accessible state we have a path leading to the state $(\varepsilon, \tilde{x}_L)$, that is every accessible state is coaccessible. This proves the first part of the lemma.

Let $(u, v)$ be a pair of words such that $u$ is a proper prefix of $X$ and $v \in A^{L-|u|}$. As we proved in the first part of the lemma, the state $(\varepsilon, uv)$ is in $Q$. Since the code is maximal, for every state in $\mathcal{T}_X$ there are exactly two out edges, one with input label 1 and the other with input label 0. Then the path starting from $(\varepsilon, uv)$ and labeled $0^{|u|}$ takes to $(u, v)$. This proves that $(u, v)$ is accessible. By the previous remark $(u, v)$ is also coaccessible. This concludes the proof.

□

Given a prefix set $X$, let us denote by $Pref_i(X)$ the set of proper prefixes of elements in $X$ of length $i$ and let us denote by $Level_i(X)$ the set of nodes in $T_X$ at level $i$. As a consequence of Lemma 4.1 we have that the number of states for the transducer associated to a maximal prefix code is given by the following:

**Theorem 4.2.** *If $X$ is a maximal prefix code then,*

$$|Q| = \sum_{i=0}^{L-1} |Pref_i(X)| \cdot 2^{L-i} = \sum_{i=0}^{L-1} |Level_i(X)| \cdot 2^{L-i}.$$

By Theorem 4.2 we can deduce an exponential lower bound in $L$ for maximal prefix codes:

**Corollary 4.3.** *If $X$ is a maximal prefix code then $|Q| \geq 2^L$.*

*Proof.* There are at least $2^L$ final states. □

From this theorem we expect that the farthest from being uniform a code is, the greatest the number of states in the correspondent transducer is.

Following Theorem 4.2, we individuate two classes of maximal codes that represent respectively the best and the worst case for the state complexity for maximal prefix codes: the uniform codes and the string-codes.

We say that a prefix code $X$ is *uniform* if all the words in $X$ have the same length $L$. A maximal uniform code whose words have length $L$ contains all the words of this length, i.e. $X = A^L$. Then for uniform maximal codes:

- $|X| = 2^L$;
- $\|X\| = L\ 2^L$;
- $|T_X| = 2^{L+1} - 1$.

Since, for each $i$, the number of nodes at level $i$ in $T_X$ is $2^i$ we get:

**Theorem 4.4.** *If $X$ is a uniform maximal code then $|Q| = L2^L$.*

**Corollary 4.5.** *The upper bound given by Theorem 3.1 is tight.*

The number of states of the transducer associated to a maximal prefix code, computed in terms of the different sizes is given by the following:

**Corollary 4.6.** *Let $X$ be a uniform maximal prefix code. The number of states of $\mathcal{T}_X$ is equal to*

- $\|X\|$
- $L2^L = |X|\log(|X|)$
- $\frac{1}{2}(\log(|T_X|+1)-1)(|T_X|+1) = O(|T_X|\log(|T_X|)$*;*

*Proof.* For uniform maximal codes, we have that $L2^L = \|X\|$.

Moreover, since $|X| = 2^L$, we have that $|Q| = |X|\log(|X|)$.

From $|T_X| = 2^{L+1} - 1$ it follows that $L = \log(|T_X|+1)/2$. Thus $|Q| = L2^L = \log((|T_X|+1)/2)(|T_X|+1)/2 = 1/2(\log(|T_X|+1)-1)(|T_X|+1)$ that is an $|Q| = O(|T_X|\log|T_X|$. □

This means that for maximal uniform codes we have a linear dependence between the size of the transducer associated to $X$ and the length of $X$. Moreover there is an almost linear dependence (with a multiplicative logarithmic factor) on the cardinality of $X$ and the size of its tree.

We have to remark that the study of the maximal uniform codes is here given for sake of completeness, since actually maximal uniform codes are bifix, so they do not need the application og Girod's coding in order to have bilateral deciphering with bounded delay.

On the other side, as one can guess, uniform maximal code are the biggest among the maximal codes having the same value of $L$, since the corresponding tree has the greatest number of nodes at a given level. Conversely, the best case happens when the number of nodes at a given level is 1, corresponding to string-codes.

Let $u$ be a word over $A = \{0, 1\}$. We define $X_u$ the *string-code* of $u$ as $X_u = \{u\} \cup \{va \mid v\bar{a} \in Pref(u)\}$, where $Pref(u)$ is the set of prefixes of $u$, and $\bar{a}$ is the opposite bit of $a$. In this case $L$ is the length of $u$.

If $X$ is a string-code, then

- $\|X\| = L(L+1)/2 + L$,
- $|X| = L + 1$ and
- $|T_X| = 2L$.

Since, for each $i$, the number of nodes at level $i$ in $T_X$ is 1 we get:

**Theorem 4.7.** *If $X_u$ is a string-code then $|Q| = 2^{L+1} - 2$.*

*Proof.* By definition $X_u$ is a maximal prefix code. Moreover for each level, there is just one internal node. By Theorem 4.2,

$$|Q| = \sum_{i=0}^{L-1} |Level_i(X)| \cdot 2^{L-i} = \sum_{i=1,L-1} 1 \cdot 2^{L-i} = \sum_{j=1,L-1} 2^j = 2^L - 2$$

□

This means that $|Q| = O(2^{\sqrt{\|X_u\|}})$, and $|Q| = O(2^{|T_{X_u}|})$ and $|Q| = O(2^{|X|})$. Thus these codes seem to have the worst behavior in terms of the number of states in relation with the different definitions of size of the code.

The theorems proved in this section formalizes somehow the intuition that the farthest a code is from being uniform and maximal, the greatest the number of states is in the correspondent transducer, depending either on $|T_X|$ or on $\|X\|$.

## 5. Uniform codes

In the previous section we have given a value for the number of states of a transducer associated to the Girod's decoding by a uniform maximal code.

In this section we consider $X$ a uniform non-maximal prefix code. This means that all the words in the code have the same length $L$, but they are not necessarily all of the $2^L$ possible words of length $L$.

For uniform codes of two words we have a result giving a precise number of the states of the transducer:

**Theorem 5.1.** *Let $X = \{x_1, x_2\}$ be a uniform code and let $u$ be the longest common prefix between $x_1$ and $x_2$. Then:*

$$|Q| = \begin{cases} |T_X| - 3|u| + 2L - 3 & \text{if } |u| < L/2 \\ |T_X| - |u| + L - 2 & \text{if } |u| \geq L/2 \end{cases}$$

*Proof.* Starting from the initial state $(\varepsilon, \tilde{x_1})$ there are two paths, the first leading to the state $(\varepsilon, \tilde{x_1})$ itself with output label $b_1$ and the second leading to the state $(\varepsilon, \tilde{x_2})$ with output label $b_2$. These two paths share the path from $(\varepsilon, \tilde{x_1})$ to the state $(u, v)$ where $u$ is the common prefix between $x_1$ and $x_2$ and $v$ is the suffix of $\tilde{x_1}$ of length $L - |u|$. The number of the states involved in these two paths is $|T_X| - 1$.

Starting from $(\varepsilon, \tilde{x_2})$ there are two paths: the first leading to $(\varepsilon, \tilde{x_1})$, with output label $b_1$, and other leading to the state $(\varepsilon, \tilde{x_2})$ itself, with output label $b_2$. As before, these two paths share a common path to the state $(u, v')$ where $u$ is the common prefix between $x_1$ and $x_2$ and $v'$ a suffix of $\tilde{v_2}$.

However, this time we do not have to count $|T_X|$ states, since some of the states needed have already been created before.

In particular, since $u$ is the common prefix between $x_1$ and $x_2$, $\tilde{u}$ is a suffix of $\tilde{x_2}$. Since there is a cycle from $(\varepsilon, \tilde{x_2})$ to itself, this path contains the state $(pref_{L-|u|}(x_2), \tilde{u})$. Analogously, since there is a path from $(\varepsilon, \tilde{x_2})$ to $(\varepsilon, \tilde{x_1})$, this path contains the state $(pref_{L-|u|}(x_1), \tilde{u})$. Both of these states are already been created in the constructions of the paths from $(\epsilon, \tilde{x_1})$, since $\tilde{u}$ is a suffix both of $\tilde{x_1}$ and $\tilde{x_2}$.

If $|u| < L/2$, then $Pref_{L-|u|}(x_1)$ and $Pref_{L-|u|}(x_2)$ are different. So the two states $(Pref_{L-|u|}(x_1), \tilde{u})$ and $(Pref_{L-|u|}(x_2), \tilde{u})$ are different too and they lie respectively in the path going from $(u, v)$ to $(\epsilon, \tilde{x_2})$, and from $(u, v)$ to $(\epsilon, \tilde{x_2})$. That is we need to add $|u| + 2(L - |u| - 1)$ states to the ones already constructed before. That is:

$$|Q| = |T_X| - 1 + |u| + 2(L - 2|u| - 1) = |T_X| - 3|u| + 2L - 3.$$

If $|u| \geq L/2$, then $Pref_{L-|u|}(x_1) = Pref_{L-|u|}(x_2)$. This means that the state $(Pref_{L-|u|}(x_1), \tilde{u}) = (Pref_{L-|u|}(x_2), \tilde{u})$ is in the path going from $(\epsilon, \tilde{x_1})$ to $(u, v)$. In this case we need to add just $L - |u| - 1$ states. Then we have:

$$|Q| = |T_X| + L - u - 2.$$

□

The following proposition establishes an upper bound of $O(|X||T_X|)$ to the number of states of a transducer associated to a non maximal uniform prefix code:

**Proposition 5.2.** *If $X = \{x_1, \ldots, x_k\}$ is a non maximal uniform prefix code then $|Q| \leq |X||T_X| - |X|^2$. This bound is tight for codes of two words beginning with different letters.*

*Proof.* Starting from initial state $(\varepsilon, \tilde{x_1})$, for each $j = 1, \cdots, k$ there must exist a path with output label $b_j$, ending in the state $(\varepsilon, \tilde{x_j})$. Notice that the final states are all of the form $(\epsilon, \tilde{x_j})$ since the code is uniform. All of these paths should follow the structure of the tree $T_X$ for what observed in the proof of the previous theorem. So the number of the states needed for these paths is $|T_X| - 1$.

From each state $(\varepsilon, \tilde{x_j})$, $j \neq 1$, there is again a path to each state $(\epsilon, \tilde{x_l})$, and all of these paths reproduce the structure of the tree. Thus, for each state $(\varepsilon, \tilde{x_j}) \neq (\varepsilon, \tilde{x_i})$, we must add at most $|T_X| - |X| - 1$ states.

Thus finally we get an upper bound for general prefix codes

$$|Q| \leq |T_X| - 1 + (|X| - 1)(|T_X| - |X| - 1) = |X||T_X| - |X|^2$$

Tightness follows by Theorem 5.1 when $u = \epsilon$. □

## 6. Conclusions and open problems

In this paper we have given some bounds to the number of states of transducers associated to the Girod's deciphering by a given prefix code, depending on different

notions of size associated to a prefix code. We have found a tight general upper bound depending on the length of the longest word in the code. Moreover we have found a precise value of the number of states for transducers associated to maximal uniform prefix codes and "string codes", depending on the different sizes of the code, that represent the best and the worst case for maximal codes, providing upper and lower bounds for maximal codes. Moreover we proved that "isomorphic" prefix codes, i.e. codes associated to isomorphic trees, have associated transducers that are isomorphic as unlabeled graphs.

Unfortunately, we do not have an exact formula giving the number of states of the transducer for any maximal code $X$, given its sizes (length of the longest word, cardinality and sum of the lengths of its elements). It remains an open question to find such a formula.

Also, we do not have a general formula, but just an upper bound, for such a number for codes that are not maximal, except for the case of a uniform code with just two words. A possible strategy could be to see how the transducer grows, in terms of number of states, when we add a new word to the prefix code, and in which measure this growth depends on how much a prefix of the new added word matches a prefix of another word already in the code.

Another open problem is the one of doing an average study of the number of states for different distributions on prefix codes, stating how big the transducers associated to a prefix code are in the average.

In conclusion, we also think that studying the transducer associated to Girod's deciphering by prefix code can also inspire some similar methodologies that can be applied not necessarily to finite prefix codes, but also to infinite prefix codes, and finite codes that are not necessarily prefix, but with some weaker property.

## 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] 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] D. Salomon. *Variable-length codes for data compression*. Springer-Verlag, 2007.

Communicated by (The editor will be set by the publisher).