ON ROTATIONALLY INVARIANT CODES

MARIE-PIERRE BÉAL*

Abstract. We give an extension of the work of Trott *et al.* on rotationally invariant codes and the work of Adler *et al.* and Marcus on constrained codes, to constrained codes invariant by a permutative map. A constructive proof is given of the existence a finite-state invertible ϕ-invariant code from arbitrary n-ary sequences to a ϕ-invariant sofic system at constant rate $1:1$ provided that ϕ is a permutative conjugacy of the sofic channel and that the Shannon capacity of the channel exceeds the capacity of the source. In the case of equality of the capacities of the source and of the ϕ-invariant channel, the same result holds when the permutative map has no bi-infinite fixed point which is a direct extension of the result of Trott *et al.* for the rotational case. The decoders can be made both sliding block, or "state independent", and ϕ-invariant.

1. Introduction. In this paper, we shall consider constrained systems modelled by finite automata. They are called sofic systems and defined as the sets of all bi-infinite sequences generated by walks on a finite directed graph labeled by symbols in a finite alphabet. The labeled graph is called an automaton. Encoding unconstrained sequences into constrained ones has many practical applications for storage devices (see for instance [19], [25]). In many cases, both encoder and decoder are finite state automata and the decoder is sliding block, that is, the output of the decoder depends only on a bounded window of the input of the decoder. Indeed, usually the encoded sequences are fed as inputs to a noisy channel and, if the decoder is sliding block, a given channel error will give rise to at a most finitely many data errors. Several general coding methods for constrained channels have been since given to solve these coding problems (see [25] and references therein). All methods build finite state invertible codes at a constant rate $p:q$ from the set of all bi-infinite sequences over a finite alphabet to a sofic system. The coding rate depends on the Shannon capacity of the channel and of the source. It can be assumed that the rate is $1:1$ after a recoding. Some constraints are called finite type constraints when they are characterized by a finite list of forbidden blocks. The bi-infinite constrained sequences are thus exactly the sequences that do not contain any element of the list as a subblock. Of special interest are the run-length sequences which are of finite type [29]. Another class of sofic constrained channel is the class of constrained systems called almost of finite (see [20], [11]). All these channels play an important role in symbolic dynamics. For practical applications, the codes are constructed on semi-infinite or one-sided sequences. But the treatment takes the bi-infinite or two sided point of view since it appears to be very natural and useful for the one sided situation.

*Institut Gaspard Monge, Université de Marne-la-Vallée, 77454 Marne-la-Vallée Cedex 2, France. beal@univ-mlv.fr.

Beside the domain of coding for constrained channels is trellis coding, mostly used to correct errors. One aspect of trellis coding that has come under increasing study is the search for codes that can handle phase rotations of a signal set. The rotations considered are those caused for instance by a demodulator in a communication system. The modulation signal set has a two dimensional rotational symmetry and the transmission system can introduce a phase ambiguity at the receiver. In such a practical system, the receiver must recover the correct phase of the signal. If phase symmetries exist, the receiver may lock to the wrong phase. Thus the code must either have no phase symmetries or be immune to any rotations resulting from these symmetries. The latter type is known as rotationally invariant codes. It is a conventional method to solve this problem. The idea behind rotationally invariant codes is to find an encoder and a decoder that ensure the following property : given any code sequence, if we rotate each symbol through a fixed rotational symmetry, then the new sequence of rotated symbols is also a valid code sequence. Moreover all rotations of a code sequence have to decode to the same source sequence. Conditions for rotational invariance and several rotationally invariant codes were described by Wei [31] in [32]. Many papers have been since devoted to code constructions. Several classes of codes have raised ; some of them use nonlinear parity check equations, linear systems over rings, algebraic systems, group systems (see [12, 13, 14]).

A basic theory of rotational invariance has been developed in [30] and [10]. It provides a mathematical foundation of rotationally invariant codes and rotationally invariant encoders which is independent from their algebraic structure. The code sequences have a finite state trellis diagram. They are thus modelled again by a sofic system. In a lot of cases, the channel is unconstrained in the sense that it is the set of all bi-infinite sequences of letters in a finite alphabet. Each letter represents a signal point in the plane. The rotation in the plane corresponds to a permutation ϕ which has no fixed point in the alphabet. This permutation is then applied to the bi-infinite sequences (letter by letter) and a channel is invariant by permutation if the permutation of any bi-infinite sequence is a code sequence. An encoder associates to each bi-infinite unconstrained input sequence an output one in the channel. The decoder is said to be invariant by rotation, or ϕ-invariant, if each rotated coded sequence has the same decoding. These rotationally invariant codes are usually finite state invertible codes at a constant rate $p : q$, with a sliding block decoder which is moreover invariant by rotation. In [30] are given necessary and sufficient conditions for codes and encoders to be rotationally invariant. A rotationally invariant code construction is also given in the case where the permutation does not fix any letter alphabet. The method uses the symbolic dynamics state splitting process which is applied to channels of finite type and represented by a graph which has a constant outdegree. In order to handle channel representations that have not a constant outdegree, the state splitting algorithm of Adler *et al.* [1] can be used to solve this question (see [30, Section IV-D-2] where the method is outlined).

The following coding theorem for constrained channels was proved in [1], where $\mathrm{cap}(S)$ denotes the Shannon capacity of a channel S.

THEOREM 1.1. *Let A and B be two finite alphabets. Let $S \subset B^{\mathbb{Z}}$ be a shift of finite type such that* $\mathrm{cap}(S) \geq \mathrm{cap}(A^{\mathbb{Z}})$. *Then there is a finite state invertible code from $A^{\mathbb{Z}}$ to S at constant rate* $1:1$ *with sliding block decoder.*

The results from [30] lead then to the following coding theorem for rotation channels.

THEOREM 1.2. *Let A and B be two finite alphabets and ϕ a permutation of B which has no fixed point in B. Let $S \subset B^{\mathbb{Z}}$ be a ϕ-invariant shift of finite type such that* $\mathrm{cap}(S) \geq \mathrm{cap}(A^{\mathbb{Z}})$. *Then there is a finite state invertible code from $A^{\mathbb{Z}}$ to S at constant rate* $1:1$ *with sliding block and ϕ-invariant decoder.*

In this paper, we extend Theorem 1.2 to ϕ-invariant constrained channels, where ϕ is a one-block map conjugacy of the channel induced by any permutation of the alphabet. We call such a map a permutative map. The construction generalizes the method presented in [30] that applies in the case where ϕ is induced by a rotation which has no fixed point in the alphabet B. As in [30], we use state splittings to construct the encoder, but in the case of such permutations, we need to "split more" the finite state machine that models the channel, and choose an order for the splittings. We also use the important notion of sharp automaton introduced in [30]. The hypothesis on ϕ is here weaker than that given in [30] since ϕ is only assumed to have no bi-infinite fixed point. Thus, our encoder construction may apply to signal sets that have a point at the origin, which is fixed by a nontrivial rotation. Nevertheless, this is only a theoretical improvement, since the requirement of no fixed point in the alphabet entails no loss of generality for rotation channels. Indeed, in the applications, it is possible to choose an alphabet B in such a way that the phase rotation induces a permutation of B that fixes no point of the alphabet. A sufficiently fine partition of the signal constellation has no cells fixed by the rotation. The construction includes the state splitting process method outlined in [30, Section IV-D-2], to get a graph which has a constant outdegree, that we extend to graphs that may have several strongly connected components. We point out that it is not possible to handle only representations that have a strongly connected graph to obtain the result.

The method does not apply to the more general case of a channel modelled by a sofic system in the case where the source and the channel have the same capacity, although some extensions to almost of finite type shifts could be considered (see the discussion at the end of the paper). In the case of sofic shifts with a capacity that strictly exceeds the capacity of the source, the following coding theorem has been obtained in [24] using results in [1] (see also [5]).

THEOREM 1.3. *Let A and B be two finite alphabets. Let $S \subset B^{\mathbb{Z}}$ be a sofic system such that* $\mathrm{cap}(S) > \mathrm{cap}(A^{\mathbb{Z}})$. *Then there exist a subshift of finite type $T \subset S$ and a finite state invertible code from $A^{\mathbb{Z}}$ to T at constant rate* $1:1$ *with sliding block decoder.*

We extend this result to ϕ-invariant sofic channels, i. e. to sofic channels invariant by a permutative conjugacy ϕ. Since we consider channels whose capacity is strictly greater than the source capacity, this allows us to completely remove the hypothesis that ϕ has no fixed point. We prove the following result.

THEOREM 1.4. *Let A and B be two finite alphabets and ϕ be a permutative map from $A^{\mathbb{Z}}$ to $B^{\mathbb{Z}}$. Let $S \subset B^{\mathbb{Z}}$ be a ϕ-invariant sofic system such that* $\mathrm{cap}(S) > \mathrm{cap}(A^{\mathbb{Z}})$ *Then there exist a subshift of finite type $T \subset S$ and a finite state invertible code from $A^{\mathbb{Z}}$ to T at constant rate* $1:1$ *with sliding block and ϕ-invariant decoder.*

The construction that we use in the proof does not use state splitting this time but the "method of poles" or "method of principal states" introduced by P. A. Franaszek and adapted to sofic channels in [8] (see also [16], [17], [18], [6], [4], [25]). It uses the notion of strongly synchronizing states of an automaton introduced in [7]. Again, we mention that is not possible to restrict us here to channels that have a presentation which is both ϕ-invariant and has a strongly connected graph.

The results presented in the paper do not provide new useful codes for rotation channels. In this framework, rotationally invariant codes appear to be a special case of a larger class that we call codes invariant by a permutation or a permutative conjugacy. Although the hypothesis on the conjugacy of being permutative is still very strong, it seems nevertheless difficult to weaken it to obtain similar results for more general conjugacies.

The paper is organized as follows. In the second section, we mention some motivations of the need of rotationally invariant codes and refer to [30] for more details. We recall some background on symbolic dynamics, automata theory, and coding theory. The terminology borrows from all these areas (see [15] for a multilingual dictionary). We give the main definitions of objects used in the paper like an automaton, a transducer, a sofic system and a shift of finite type, the operation of state splitting in an automaton. We describe the rotation and the permutation channel. Such a channel is called ϕ-invariant. Several examples of such channels are given. We mention the test obtained in [30] to check whether a sofic code is invariant by a permutation. Section 3 is devoted to the construction of coding schemes adapted to ϕ-invariant channels where ϕ is a permutative conjugacy. We prove Theorems 1.2 and 1.4. We build a transducer with a deterministic input automaton which recognizes the source sequences in a k-letter alphabet, and with a local (or definite) output automaton that recognizes the code sequences. The decoder is a sliding block decoder and hence, it has a limited error propagation. An example of code construction is given which illustrates the proof of Theorem 1.2. Some possible extensions of the results are discussed at the end of the paper.

2. Definitions and background.

2.1. The rotation channel. Rotationally invariant trellis codes apply to overcome the phase ambiguity in some kinds of modulation channels. Let us consider for example a two-dimensional discrete input sequence $((a_k, b_k))_{k\in\mathbb{Z}}$ generated by a source. It is separated into two one dimensional sequences and low-pass filtered. The resulting signals $A(t)$ and $B(t)$ corresponding to one pair (a_k, b_k) are modulated, summed and transmitted over a noisy channel. The transmitted signal is

$$s(t) = A(t)\cos(2\pi f_c t) + B(t)\sin(2\pi f_c t).$$

Due to the propagation delay and in the case where there is no noise, the received signal $r(t)$ can have the following form

$$r(t) = A(t)\cos(2\pi f_c t + \phi) + B(t)\sin(2\pi f_c t + \phi).$$

At the reception the carrier phase ϕ is estimated before the amplitude. It comes out that the carrier phase is estimated up to an ambiguity of $360°/M$, where M is a constant which depends of the signal constellation. This phase error is due to the method used to recover the phase which is sometimes based on networks that first kill the carrier information. We shall not discuss here these techniques and refer to [28] for a description of demodulation schemes. The phase recovered is denoted by $\hat{\phi}$ and the *phase error* is defined as

$$\phi_e = \phi - \hat{\phi}.$$

One then multiplies the received signal $r(t)$ by $\cos(2\pi f_c t+\hat{\phi})$ and by $\sin(2\pi f_c t+\hat{\phi})$. The components of double frequency are eliminated to give the estimations $\hat{A}(t)$ and $\hat{B}(t)$ of $A(t)$ and $B(t)$ respectively.

One easily get, by the use of some elementary trigonometric equalities,

$$\begin{aligned}\hat{A}(t) &= A(t)\cos\phi_e - B(t)\sin\phi_e\\ \hat{B}(t) &= A(t)\sin\phi_e + B(t)\cos\phi_e.\end{aligned}$$

Hence, if $(\hat{a}_k, \hat{b}_k)$ is the two dimensional output signal point recombined after demodulation, we get

$$\begin{pmatrix}\hat{a}_k\\ \hat{b}_k\end{pmatrix} = \begin{pmatrix}\cos\phi_e & -\sin\phi_e\\ \sin\phi_e & \cos\phi_e\end{pmatrix}\begin{pmatrix}a_k\\ b_k\end{pmatrix}.$$

Thus the input signal (a_k, b_k) is rotated in the plane by the phase error ϕ_e.

We consider the set of all possible signal sequences and call it the *channel S*. One way to solve the carrier phase recovery problem is to encode an unconstrained sequence into a sequence of S, where S is supposed to be invariant by the rotation ρ by ϕ_e, and then by the group generated by the rotation. The decoding scheme must then associate to all rotations of a code sequence the same input sequence. In a sense, the phase ambiguity is not solved by the receiver. The problem is addressed to the encoder and the decoder simply ignores the phase error instead of trying to learn and correct it.

A channel which is invariant by a fixed rotation is called a *rotationally invariant system or channel.* A coding scheme such as one described above is called a *rotationally invariant encoder*. The channel model that we shall use is a discrete memoryless and two dimensional channel. It is corrupted by a fixed rotation ρ that generates the rotation group of the system. We call channels of that type *rotation channels.*

Let us consider for example the 4-PSK signal set or QPSK signal set (Quadrature Phase Shift Keying) shown in Figure 2.1. The code is the set of all sequences on the alphabet $\{0, 1, 2, 3\}$. It is invariant by the rotation group generated by a rotation ρ about the origin point by $90°$.

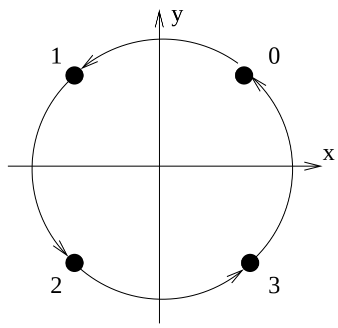

FIG. 2.1. *A 4-PSK constellation*

Trellis codes are usually described in terms of partitioning a signal set. We shall denote by A such a partition, whose elements are called the cells. In this paper, we shall not be interested in the fine structure of the cells and the symbol A simply denotes a label alphabet whose elements are called letters.

When the group of transformations is no more a rotation group, but more generally a permutation group on signal points, we speak of a channel or a system *invariant by permutation*, and of a *permutation channel.* The signal system is modelled by a finite state machine. From the point of view of symbolic dynamics, the channel is a sofic system.

2.2. Shifts invariant by permutation. In the sequel, A will be the alphabet composed of signal points. The set of bi-infinite sequences of points of A is denoted by $A^{\mathbb{Z}}$. A bi-infinite sequence

$$\dots www.www \dots,$$

obtained by a bi-infinite concatenation of a finite word w and where the dot is before the letter of index 0, is denoted by ${}^{\omega}w^{\omega}$.

A *finite automaton* $\mathcal{A} = (Q, E)$ is composed of a finite set of states Q and a finite set of edges E labeled in A. Thus $E \subset Q \times A \times Q$. The code, denoted by S, is a *sofic system*, if it is the set of bi-infinite sequences that are the labels of bi-infinite paths of a finite automaton. The sofic system is said to be recognized by the automaton $\mathcal{A}$.

The *capacity* (or *topological entropy*) of a subshift $S \subset A^{\mathbb{Z}}$ is defined as

$$\operatorname{cap}(S) = \lim_{n \to \infty} \frac{1}{n} \log \operatorname{card}(S_n \cap A^n),$$

where S_n denotes the set of finite blocks of length n that are factors of bi-infinite words of S. It is known that the capacity of a sofic channel is computable from an unambiguous (or lossless) automaton which recognizes the channel (see for instance [23]).

A finite automaton is said to be *deterministic*, or *right-resolving*, if there is at most one edge going out from a given state and with a given label. It is *codeterministic*, or *left-resolving*, if the automaton obtained by reversing the sense of the edges is deterministic. It is *deterministic with a finite delay d*, or *right-closing*, if whenever two paths of length $d+1$ start at the same state and have the same label, then they must have the same initial state. A finite automaton is said to be (m, a)-*local* (or (m, a)-*definite*) if there are two nonnegative integers m and a, such that whenever two finite paths of length $m + a$, $((p_i, a_i, p_{i+1}))_{0 \le i \le (m+a)}$ and $((p'_i, a_i, p'_{i+1}))_{0 \le i \le (m+a)}$ have the same label, then $p_m = p'_m$. An automaton is *local* if it is (m, a)-local for some integers m and a (m is for memory and a for anticipation). Remark that if an automaton is (m, a)-local, it is also $(m, a+1)$-local and $(m+1, a)$-local. A deterministic automaton is local if and only if the previous condition is satisfied with a null anticipation. It is then said to be m-local.

The following known property gives a polynomial time decision procedure to check whether a finite automaton is local (see for instance [7, 23, 9]).

Proposition 2.1. *Let $\mathcal{A}$ be an automaton which has a strongly connected graph or which is unambiguous. The two following properties are equivalent.*

(i) *the automaton $\mathcal{A}$ is local.*

(ii) *the automaton $\mathcal{A}$ has at most one cycle with a given label.*

A sofic system which can be recognized by a local automaton is called a *subshift of finite type*. A sofic system which can be recognized by an automaton with a strongly connected graph is called an *irreducible subshift*.

When the channel is not irreducible, one generally extracts a strongly connected component which constitutes a new channel of same capacity which is irreducible. The finite type condition appears to be satisfied in lot of applications.

We recall the notion of strongly synchronizing states introduced in [7]. Let m and a be two nonnegative integers. For each state p of an automaton $\mathcal{A}$, we define the set $E_p^{(m,a)}$ as the set of finite words uv, where u is a word of length m which is the label of a path terminating at p, and v is a word of length a which is the label of a path starting at p.

A state p of an automaton $\mathcal{A}$ is said to be (m,a)-*strongly synchronizing* if $E_p^{(m,a)} \cap E_q^{(m,a)} = \emptyset$ for any state $q \neq p$. A state is said to be *strongly synchronizing* if it is (m,a)-strongly synchronizing for some integers m and a. A state p of a deterministic automaton is (m,a)-strongly synchronizing if and only if it is $(m,0)$-strongly synchronizing.

A state p of an automaton $\mathcal{A}$ is strongly synchronizing if and only there are not two distinct equally labeled bi-infinite paths such that the first one goes through the state p at some index i, and the second one goes through a state $q \neq p$ at the same index. This property is computable in a polynomial time in the number of states of $\mathcal{A}$ (see [7] page 72). An automaton is local if and only if all its states are strongly synchronizing.

A *synchronizing word*, or *reset sequence*, or *magic word*, of an automaton is a finite word uv, where u is a word of length m and v a word of length a, such that all paths labeled by uv in the automaton go through the same state at the index m. All words in the set $E_p^{(m,a)}$ of an (m,a)-strongly synchronizing state are synchronizing words.

The word b for instance is a synchronizing word of the automaton of Figure 2.2. It focuses to state 1. But none state of this automaton is strongly synchronizing.

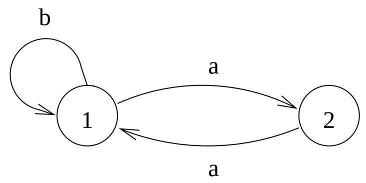

FIG. 2.2. *An automaton without any strongly synchronizing state*

We now come to the definition of conjugacies. Let A and B be two finite alphabets. A function f from $A^{\mathbb{Z}}$ to $B^{\mathbb{Z}}$ is a *sliding block map* if there are nonnegative integers m, a, and a function $\bar{f}$ from A^l to B, where $l = m+a+1$, such that for all $\mathbf{u} \in A^{\mathbb{Z}}$, the image $\mathbf{v} = f(\mathbf{u})$ of $\mathbf{u}$ is the bi-infinite word of $B^{\mathbb{Z}}$ defined by $v_n = \bar{f}(u_{n-m} \cdots u_{n+a})$ for all $n \in \mathbb{Z}$. Thus the letter v_n only depends on the finite subblock $u_{n-m} \cdots u_{n+a}$ of $\mathbf{u}$. The integer $l = m+a+1$

is called the size of the so called sliding window. We shall say that f is a (m, a)-sliding block map. The integer m is called the *memory* and a the *anticipation* of the sliding block map. A sliding block map f is said to have memory m and anticipation a if there is a map $\bar{f}$ defined as above. Note that m and a depend on the choice of $\bar{f}$ and may not be unique.

A *conjugacy* between a subshift $S \subset A^{\mathbb{Z}}$ and a subshift $T \subset B^{\mathbb{Z}}$ is a bijective sliding block map. Its inverse is also a sliding block map, sometimes with a different sliding window size. A sliding block window is represented in Figure 2.3.

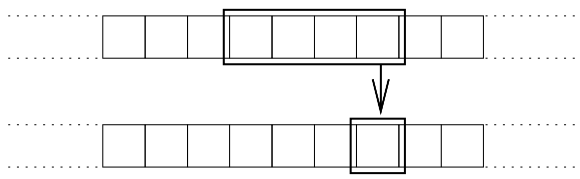

FIG. 2.3. *A sliding block window*

A sliding block map f is said to be *right-resolving* if there is a nonnegative integer m and a function $\hat{f}$ from $A^{(2m+1)}$ to B such that for all $\mathbf{u} \in A^{\mathbb{Z}}$, if $\mathbf{v} = f(\mathbf{u})$ then $u_n = \hat{f}(u_{n-m} \dots u_{n-1} v_{n-m} \dots v_{n-1} v_n)$. It is actually a property of the inverse of the map f. It is *right-closing* if there are nonnegative integers m, a and a function $\hat{f}$ from $A^{(2m+a+1)}$ to B such that for all $\mathbf{u} \in A^{\mathbb{Z}}$, if $\mathbf{v} = f(\mathbf{u})$ then $u_n = \hat{f}(u_{n-m} \dots u_{n-1} v_{n-m} \dots v_{n+a}))$.

Let π be a permutation of the alphabet A, and Π the group of permutations generated by π. The permutation π on A induces a conjugacy ϕ from $A^{\mathbb{Z}}$ to $A^{\mathbb{Z}}$ defined by

$$\phi((a_k)_{k \in \mathbb{Z}}) = (\pi(a_k))_{k \in \mathbb{Z}}.$$

If $u = a_1 \dots a_n$ is a finite word of A^*, we also denote by $\phi(u)$ the word $\pi(a_1) \dots \pi(a_n)$. Such a conjugacy is called a *permutative conjugacy*. The subshift or channel is *invariant by permutation* or *ϕ-invariant* if $\phi(S) = S$. The conjugacy ϕ and its inverse are sliding block maps with no memory and no anticipation.

Let f be a conjugacy of a subshift S with memory m and anticipation a. Let $u = u_{i-m} \dots u_i u_{i+1} u_{i+a+r}$ be a finite word of length $m + a + r$, where r is a nonnegative integer. We denote by $f(u)$ the finite word $v = v_1 \dots v_r$ of length r defined by $v_i = \bar{f}(u_{i-m} \dots u_i u_{i+1} u_{i+a})$.

As an example, the rotation about the origin by 90° on the 4-PSK signal of Figure 2.1 induces the permutation $\pi = (1230)$, expressed in the standard cycle notation, on the alphabet $A = \{0, 1, 2, 3\}$. The group Π is $\{\mathrm{Id}, \pi, \pi^2, \pi^3\}$, where Id denotes the identity map on A. In this example, the channel is the set of all bi-infinite paths of $A^{\mathbb{Z}}$. It is called the full shift on a 4-letter alphabet.

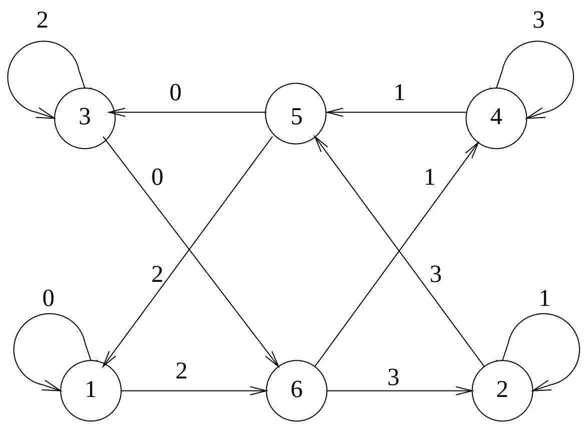

FIG. 2.4. *Oerder's 6-state m-PSK code*

The example of Figure 2.4 is Oerder's 6-state m-PSK code (see [30] and [27]). It is a 4-PSK code with label alphabet $A = \{0, 1, 2, 3\}$. The 90° rotation induces the circular permutation (0123) on A. This rotation induces in turn the permutation (1234)(56) on graph nodes of the automaton of Figure 2.4. This means that if a sequence $\mathbf{u}$ in the channel is the label of a path going through state q at time t, the image $\pi(\mathbf{u})$ of $\mathbf{u}$ is a code sequence which is the label of a path going through the state $\pi(q)$ at the same time.

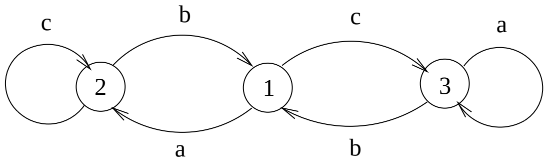

FIG. 2.5. *A channel invariant by permutation*

Another example of shift of a finite type S invariant by permutation is the shift recognized by the automaton of Figure 2.5. It is invariant by the permutation $\pi = (ac)(b)$ on $A = \{a, b, c\}$. This permutation has a fixed point in A, the point b, but has no fixed channel sequence since ${}^\omega b^\omega$ does not belong to S.

The shift S given by Figure 2.6 is a shift of finite type invariant by permutation. It is invariant by the permutation $\pi = (ac)(b)(d)$ on $A = \{a, b, c, d\}$. This permutation has a fixed point in A, the point b, and it has a fixed channel sequence ${}^\omega b^\omega$.

It is known that any irreducible sofic shift has a minimal representation, that is a minimal finite state machine which recognizes it (see for instance [7], [23], [9]). This minimal representation is a deterministic finite automaton

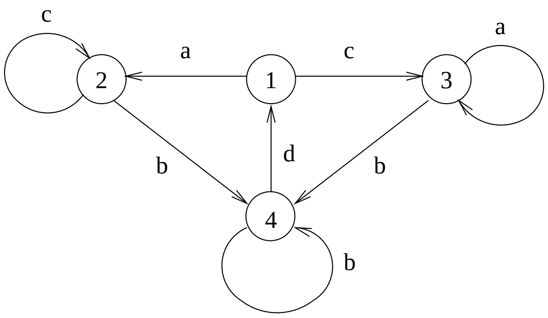

FIG. 2.6. *Another channel invariant by permutation*

called the Fischer cover of the sofic system. The sofic shift is of finite type if and only if its minimal automaton is local.

Let π be a fixed permutation of A. Let S be an irreducible sofic shift. We define the image of S by the permutation π by $\pi(S) = \{\pi(\mathbf{u}), \mathbf{u} \in S\}$. We define the *image of an automaton* $\mathcal{A} = (Q, E)$ by π as the automaton $\pi(\mathcal{A}) = (Q, \pi(E))$, where $\pi(E) = \{(p, \pi(a), q) \mid (p, a, q) \in E\}$. If S is recognized by an automaton $\mathcal{A} = (Q, E)$, then $\pi(S)$ is recognized by $\pi(\mathcal{A})$.

PROPOSITION 2.2 ([30]). *An irreducible sofic shift, whose minimal automaton is $\mathcal{A}$, is invariant by a permutation π if and only if, up to an isomorphism, $\mathcal{A} = \pi(\mathcal{A})$.*

2.3. State splitting. In this subsection, we recall a well known construction on automata and graphs that is the state splitting process (see [1, 23, 22, 25, 26, 7]).

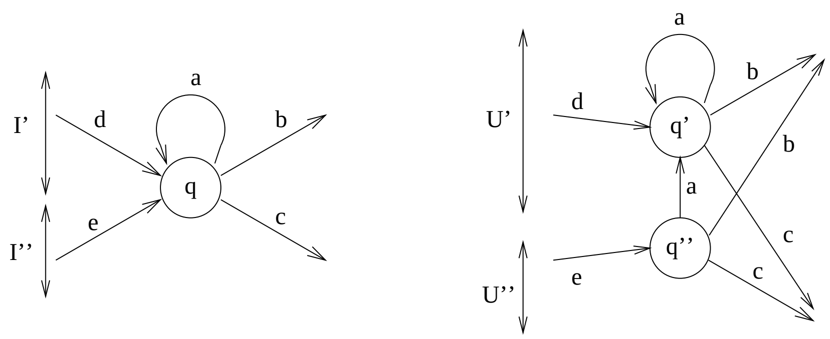

FIG. 2.7. *An input state splitting*

We define the operation of *input state splitting* in an automaton $\mathcal{A} = (Q, E)$. Let q be a vertex of Q and let I be the set of edges coming in q. Let $I = I' + I''$ be a partition of I. The operation of *input state splitting* relative to (I', I'') transforms $\mathcal{A}$ into the automaton $\mathcal{B} = (Q', E')$, where

$Q' = (Q \setminus \{q\}) \cup q' \cup q''$ is obtained from Q by splitting state q into two states q' and q'', and where E' is defined as follows.

(i) All edges of E that are not incident to q are left unchanged.

(ii) Both states q' and q'' have the same output edges as q.

(iii) The input edges of q are distributed between q' and q'' according to the partition of I into I' and I''. We denote U' and U'' the sets of input edges of q' and q'' respectively. Thus $U' = \{(p, x, q') \mid (p, x, q) \in I'\}$ and $U'' = \{(p, x, q'') \mid (p, x, q) \in I''\}$.

An input state splitting transforms a deterministic automaton in a deterministic automaton. The notion of *output state splitting* is defined similarly [23].

A state splitting induces a conjugacy between the bi-infinite paths of an automaton and the bi-infinite paths of the split automaton. The split automaton recognizes the same sofic shift as the initial one. State splitting keeps the locality of an automaton. More precisely, we have the following known properties (see for example [23] or [7]).

PROPOSITION 2.3. *An input state splitting transforms a (m, a)-local automaton in a $(m+1, a)$-local automaton. An output state splitting transforms a (m, a)-local automaton in a $(m, a+1)$-local automaton.*

3. A coding scheme for the ϕ-invariant channel.

3.1. Transducers and coding/decoding schemes. In this section A and B are finite alphabets. We consider an irreducible sofic channel $S \subset B^{\mathbb{Z}}$ and ϕ a conjugacy of S. Such a channel is ϕ-invariant.

A *transducer* will be here an automaton labeled in $A \times B$. The input (resp. output) automaton of the transducer is the automaton obtained after discarding the second (resp. first) component of labels of edges of the transducer.

The coding/decoding schemes that we shall use are modelled by transducers which have a complete deterministic input automaton which recognizes the full shift on a k-letter alphabet, and a local output automaton which recognizes a ϕ-invariant subshift of finite type T of the channel S. The coding scheme consists in choosing an initial state in the transducer and associating to each sequence of source symbols the output label of the unique path that begins at the initial state and which is input-labeled by the source sequence. The coding is thus sequential since the transducer has a deterministic input automaton. Since the output of the transducer is local, each bi-infinite word of T is the label of a unique path of the output of the transducer. We then decode it by the input label of this path. It is well known that the decoding function d is a *sliding block map* (see for instance [1], [23], [7], [9]). Such a decoding scheme is implemented by a purely combinatorial network and avoids error propagation.

The decoder of a coding/decoding scheme (or a transducer) is said to be a ϕ-*invariant* if and only if it is a sliding block map such that, moreover, for any $\mathbf{u} \in S$,

$$d(\mathbf{u}) = d(\phi(\mathbf{u})).$$

.

We say that a sliding block map ϕ has no fixed channel sequence if $\phi(\mathbf{u}) \neq \mathbf{u}$ for any $\mathbf{u} \in S$. If ϕ is conjugacy from $B^{\mathbb{Z}}$ to $B^{\mathbb{Z}}$ such that $\phi(S) = S$, we define I as

$$I = \{\mathbf{u} \in S \mid \phi(\mathbf{u}) = \mathbf{u}\}.$$

Thus ϕ has no fixed point if and only if I is empty. The set I can be computed as follows. Let us assume that ϕ has memory m and anticipation a. Let F be the set of blocks $u = u_{-m} \dots u_a$ of length $m + a + 1$ such that $u_0 = \phi(u)$. Let X be set of bi-infinite sequences such that all finite factors of length $m + a + 1$ are in X. Thus X is subshift of finite type and $S \cap X$ is a computable sofic shift equal to I. In the case where ϕ is induced by a permutation π of the finite alphabet B, an automaton recognizing I is obtained from an automaton recognizing S by discarding all edges labeled by letters fixed by π.

The permutation π in Example 2.2 for instance has no fixed channel sequence but has a fixed point in the alphabet. The permutation π in Example 2.2 has a fixed point in the alphabet.

3.2. Permutation induced on the states. Let $\mathcal{A}$ by a finite automaton which recognizes a sofic shift $S \subset B^{\mathbb{Z}}$. Let ϕ be a sliding block map from $B^{\mathbb{Z}}$ to $B^{\mathbb{Z}}$ such that $\phi(S) = S$. For any state q of $\mathcal{A}$, we define the set of bi-infinite words E_q as the set of labels of bi-infinite paths $(q_i, b_i, q_{i+1})_{i \in \mathbb{Z}}$ such that $q_0 = q$. We denote by $\phi(E_q)$ the set of bi-infinite words $\phi(\mathbf{u})$ for all $\mathbf{u} \in E_q$. The sliding block map ϕ *induces a permutation of the states* of the automaton if and only if there is a permutation π of the states of the automaton such that, for any state q,

$$\phi(E_q) = E_{\pi(q)}.$$

We consider the case where the automaton $\mathcal{A}$ is local and where ϕ is a conjugacy of S. Let us assume that ϕ has memory m and anticipation a. Since all states of $\mathcal{A}$ are strongly synchronizing, there are integers m', a' such that $E_p^{(m',a')} \cap E_q^{(m',a')} = \emptyset$ for any states $p \neq q$. Then ϕ induces a permutation π on the states of $\mathcal{A}$ if and only if, for any state q, $\phi(E_q^{(m+m',a+a')}) = E_{\pi(q)}^{(m',a')}$.

It is not always true that a sliding block map ϕ induces a permutation of the states of a local automaton even if ϕ is a conjugacy induced by a permutation of the letters of the alphabet. Consider for example the local automaton of Figure 3.1. It recognizes the same shift of finite type as the shift of Figure 2.5, which is invariant by permutation. For each state q, the set $E_q^{(2,0)}$ is represented inside the state himself in the figure. The permutation $\pi = (ac)(b)$ of $\{a, b, c\}$ does not induce a permutation of the set of states.

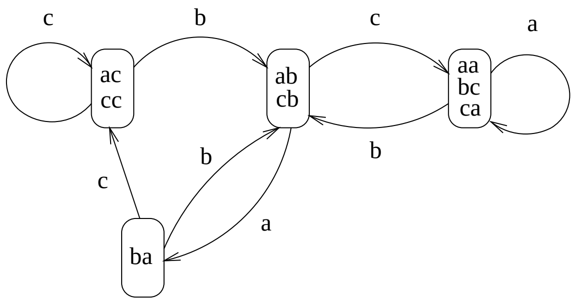

FIG. 3.1. *The permutation $\pi = (ac)(b)$ does not permute the states*

Note that if ϕ induces a permutation of the states of an automaton, it is in general not true that this property holds for a split automaton.

3.3. The sharpness condition. We now define the notion of a sharp state of an automaton relatively to a sliding block map ϕ. The notion of sharpness for an automaton was introduced in [30] in the context of rotation channels.

Let $\mathcal{A}$ by a finite automaton which recognizes a sofic shift $S \subset B^{\mathbb{Z}}$. Let ϕ be a sliding block map from $B^{\mathbb{Z}}$ to $B^{\mathbb{Z}}$ such that $\phi(S) = S$. For any state q, let E_q be the set of labels of bi-infinite paths $(q_i, b_i, q_{i+1})_{i \in \mathbb{Z}}$ such that $q_0 = q$. A state p is said to be *sharp* if

$$\phi(E_p) \cap E_p = \emptyset.$$

The automaton itself is said to be *sharp* if all its states are sharp.

In the case where the automaton $\mathcal{A}$ is (m', a')-local and where ϕ a conjugacy of S of memory m and anticipation a, a state p is sharp if and only if $\phi(E_p^{(m+m', a+a')}) \cap E_p^{(m', a')} = \emptyset$. If ϕ induces a permutation π on the states of the automaton, the automaton is sharp if and only if π has no fixed point on the states.

The sharpness condition appears to be a necessary property of the output automaton of a ϕ-invariant transducer that models a sequential encoder with a sliding block decoder. The following proposition appears in [30] in the context of rotationally invariant encoders. We give a proof of the proposition adapted to ϕ-invariant channels.

PROPOSITION 3.1 ([30]). *A ϕ-invariant transducer with a deterministic and strongly connected input automata, and where ϕ is not the identity map, has a sharp output automaton*

Proof. Let $\mathcal{T}$ be a transducer whose output automaton $\mathcal{A}$ is (m', a')-local and strongly connected. Let ϕ be a conjugacy with memory m and anticipation a such that $\mathcal{T}$ is ϕ-invariant. We define for each state q the set E_q of bi-infinite sequences which are labels of bi-infinite paths of $\mathcal{A}$ going through q at time 0.

Let us assume that the automaton $\mathcal{A}$ is not sharp. Then there is a state p such that $\phi(E_p) \cap E_p \neq \emptyset$. Let $\mathbf{x}$ be a bi-infinite sequence such that $\mathbf{x}$ and $\phi(\mathbf{x})$ belong to E_p. Let $u = x_{-(m+m')} \dots x_{-1}$ and $v = x_0 \dots x_{(a+a'-1)}$. The word uv is the label of the finite path $c = ((p_i, x_i, p_{i+1}))_{-(m+m') \leq i \leq (a+a'-1)}$ of length $m + m' + a' + a$ going through p at time 0. Let $\mathbf{v}$ be the label of a bi-infinite path C of the automaton that extends the finite path c. Thus $\phi(\mathbf{v})$ is the label of a bi-infinite path C' that goes also through the state p at time 0. We denote by R (resp. R') the right-infinite part of nonnegative indices of C (resp. C'). As the transducer is ϕ-invariant, the input labels of the edges along the bi-infinite paths C and C' are the same. Since the input automaton of the transducer is deterministic, the right-infinite paths R and R' starting at p are equal. The strongly connectivity of the graph of the transducer implies then that ϕ is the identity on the shift of finite type recognized by $\mathcal{A}$. □

Note that the proposition still holds if the input automaton of the transducer is only deterministic with a finite delay.

3.4. Construction of a ϕ-invariant transducer. A general coding scheme into finite type constrained channels has been obtained in [1] by the use of state splittings. The corresponding theorem can be stated as follows (see also [23] or [25, p. 1687]).

THEOREM 3.2 (Adler, Coppersmith and Hassner [1]). *Let A and B be two finite alphabets. Let $S \subset B^{\mathbb{Z}}$ be a shift of finite such that* $\mathrm{cap}(S) \geq \mathrm{cap}(A^{\mathbb{Z}})$. *Then there exist a subshift of finite type $T \subset S$ and a right-resolving and onto sliding block map d from T to $A^{\mathbb{Z}}$.*

The proof of the theorem provides a sequential encoder and a sliding block decoder to encode any sequence of letters of A into the constrained system defined by T.

We extend Theorem 3.2 to ϕ-invariant constrained channels, where ϕ is a one-block map conjugacy of the channel induced by a permutation, as follows.

THEOREM 3.3. *Let A and B be two finite alphabets. Let $S \subset B^{\mathbb{Z}}$ be a shift of finite such that* $\mathrm{cap}(S) \geq \mathrm{cap}(A^{\mathbb{Z}})$ *and ϕ be a permutative conjugacy of S which has no fixed point in S. Then there exist a subshift of finite type $T \subset S$ and a right-resolving and onto sliding block map d from T to $A^{\mathbb{Z}}$ such that $d \circ \phi = d$.*

Our construction extends a method presented in [30] that applies in the case where ϕ is induced by a rotation, that is a permutation which has no fixed point in the alphabet B. The hypothesis on ϕ is here weaker since ϕ is only assumed to have no bi-infinite fixed point. Thus, our encoder construction may apply to signal sets that have a point at the origin, which is fixed by a nontrivial rotation. As already mentioned in the introduction, this is only a slight theoretical improvement since the requirement of no fixed point in the alphabet entails no loss of generality for rotation channels.

We first give a lemma used in the proof of Theorem 3.3 which extends to permutation the construction given in [30].

LEMMA 3.4. *Let $S \subset B^{\mathbb{Z}}$ be a shift of finite type such that* $\mathrm{cap}(S) \geq \mathrm{cap}(A^{\mathbb{Z}})$ *and ϕ a permutative conjugacy of S which has no fixed point. Then S is recognized by a deterministic local automaton such that ϕ induces a permutation of the states of the automaton which has no fixed state.*

Proof. Let $\mathcal{A}$ be a deterministic local automaton recognizing S whose set of states is Q. Thus all states of $\mathcal{A}$ are $(m, 0)$-strongly synchronizing for some nonnegative integer m. If ϕ has no fixed point, there exists, by a compacity argument, a nonnegative integer l such that, for any word u of length l, $\phi(u) \neq u$. Without loss of generality, we choose $m \geq l$.

For each state q, we denote by P_q the set of left-infinite labels of left-infinite paths ending at q and by P_q^m the set of finite words u of length m which are labels of paths ending at p. Since all states are $(m, 0)$-strongly synchronizing, we have for any states $p \neq q$,

$$P_p^m \cap P_q^m = \emptyset.$$

For each state q, we define the finite word z_q as the longest common suffix of words in P_q^m. The word z_q may thus be empty. We choose a state p such that z_p has a minimal length among all words z_q, for $q \in Q$. If the length of z_p is strictly less than m, the set P_p^m contains two words uaz_q and $u'bz_q$, where v, v' are words, and a, b are two distinct letters of the alphabet.

If z_p is not the empty word, let c be its last letter and x the word defined by $z_p = xc$. We do an input state splitting of the state p by partitioning the incoming edges of p in the ones beginning in a state q such that bx is a suffix of z_q and the other ones. If z_p is the empty word, we do an input state splitting of the state p by partitioning the incoming edges of p in the ones labeled by a and the other ones. The state p is split in states p_1 and p_2. We get $P_{p_1}^m = P_p^m \cap A^* b z_p$ and $P_{p_2}^m = P_p^m - P_{p_1}^m$. Hence $(P_{p_1}^m, P_{p_2}^m)$ is a partition of P_p^m. In general, the automaton obtained after an input state splitting of a m-local deterministic automaton is a $(m+1)$-local one. With this kind of input state splitting, the automaton that we get is still m-local since $(P_{p_1}^m, P_{p_2}^m)$ is a partition of P_p^m. It follows that all states of the split automaton are still $(m, 0)$-strongly synchronizing. Note that the cardinalities of $P_{p_1}^m$ and $P_{p_2}^m$ are strictly less than the cardinality of P_p^m.

By iterating this input state splitting process, we get an deterministic local automaton such that for each state q, the state P_q^m is a singleton. For this automaton, each z_q is the unique word of length m of P_q^m. Thus, the words z_q for this final automaton $\mathcal{A}$ satisfy the following conditions.

(i) each left-infinite path ending at q has a label suffixed by z_q.

(ii) z_q is not a suffix of $z_{q'}$ for any $q' \neq q$.

(iii) z_q is not constant by ϕ.

We now prove that ϕ induces a permutation of $\mathcal{A}$. For any state q, we define $\pi(q)$ as the unique state such that

$$z_{\pi(q)} = \phi(z_q).$$

The existence of $\pi(q)$ is due to the fact that there is at least one bi-infinite path extending a finite path of label z_q ending at q. The unicity is due to the fact that each state is $(m, 0)$-strongly synchronizing. By considering the inverse of ϕ, it is clear that π is onto and is thus a permutation of the set of states of $\mathcal{A}$. Since for any state q, z_q is not constant by ϕ, π has no fixed point. □

It is also possible to get a codeterministic local automaton that has the same property by doing output state splittings.

We now prove Theorem 3.3. The proof uses 3.4 and includes the state splitting process outlined in [30, Section IV-D-2]. Since the representations of the channels are not supposed to have a strongly connected graph, we make the splitting process run on several strongly connected components.

We need some notation to give the proof. If $\mathbf{u}$ and $\mathbf{v}$ are vectors, $\min\{\mathbf{u}, \mathbf{v}\}$ denotes the componentwise minimum of $\mathbf{u}$ and $\mathbf{v}$. For a real number s, let $\lfloor s \rfloor$ denote the integer floor of s.

Proof. [of Theorem 3.3] The proof uses Lemma 3.4 and Theorem 3.2. By Lemma 3.4, we obtain a representation of S such that ϕ induces a permutation of states. If $\text{cap}(S) \geq \text{cap}(A^{\mathbb{Z}}) = \log(k)$, we obtain by the proof of Theorem 3.2, and by the use of output state splittings, an automaton recognizing the channel S whose graph has a minimum outdegree at least k. We would like that both properties hold for a same local representation. Since an output state splitting does not keep in general the property of inducing a permutation of the states, we have to show that the output state splitting process of [1] can be adapted to guarantee this property.

Let $\mathcal{A} = (Q, E)$ be the deterministic local automaton computed in Lemma 3.4. Let us assume that ϕ is induced by a permutation π of the letters of the alphabet. The permutative conjugacy ϕ permutes the states of $\mathcal{A}$ and we still denote by the same symbol π this permutation. Then ϕ induces a conjugacy between the bi-infinite paths of $\mathcal{A}$. For any bi-infinite path $C = ((p_i, a_i, p_{i+1})_{i \in \mathbb{Z}}$, the image of C by ϕ is defined by $\phi(C) = ((\pi(p_i), \pi(a_i), \pi(p_{i+1}))_{i \in \mathbb{Z}}$. Hence ϕ induces a permutation of the edges E of $\mathcal{A}$. If $e = (p, a, q)$ is an edge in E, we define $\pi(e) = (\pi(p), \pi(a), \pi(q))$. Finally, ϕ defines an automorphism of the automaton $\mathcal{A}$. Let $\mathcal{C}$ be a strongly connected component of $\mathcal{A}$ which has a maximal capacity. We restrict the automaton $\mathcal{A}$ to its part composed of the disjoint union of the sub-automata $\pi^n(\mathcal{C})$, for all $n \geq 0$. Note that this new automaton may be composed of several strongly connected components which are not connected and have the same capacity as the full code. It recognizes a subshift of finite of S which has the same capacity as S.

The output state splittings of [1] are guided by a positive k-approximate eigenvector of the adjacency matrix M of the graph of $\mathcal{A}$, that is a vector $\mathbf{v}$ such that $M\mathbf{v} \geq k\mathbf{v}$. Since the strongly connected components of $\mathcal{A}$ have the same capacity which is greater than $\log(k)$, the automaton admits a positive k-approximate eigenvector which can be computed by the use of Franaszek's algorithm [23, p. 153] which catches a positive approximate eigenvector with smallest maximal entry. For a great enough positive integer values r, it can be obtained as the fixed point of the sequence of column vectors $(\mathbf{v}^{(n)})_{n\geq 0}$ indexed by Q, where $\mathbf{v}^{(0)} = (r, r, \ldots, r)^{\top}$ and $\mathbf{v}^{(n+1)} = \min\{\mathbf{v}^{(n)}, \lfloor \frac{1}{k} M\mathbf{v}^{(n)} \rfloor\}$. Since for any nonnegative integer n, $v_q^{(n)} = v_{\pi(q)}^{(n)}$, the automaton $\mathcal{A}$ admits a positive k-approximate eigenvector $\mathbf{v}$ such that for each index state q, $v_q = v_{\pi(q)}$. In the case where all components of $\mathbf{v}$ are non equal, it is shown in [1] that one can split a state p in two states p_1 and p_2, according to a partition (O_1, O_2) of the output edges of p in $\mathcal{A}$, in such a way that there is a k-approximate eigenvector $\mathbf{v}'$ of the split automaton such that $v'_{p_1} + v'_{p_2} = v_p$ and $v'_q = v_q$ for $q \neq p_1, p_2$. Since the choice of p and (O_1, O_2) is governed by the approximate eigenvector, for any positive integer n, $(\pi^n(O_1), \pi^n(O_2))$ is a partition of the outgoing edges of $\pi^n(p)$ in $\mathcal{A}$ which allows a splitting of the state $\pi^n(p)$ with the same properties. It is then possible to split simultaneously each state $\pi^n(p)$ in two states $\pi^n(p)_1$ and $\pi^n(p)_2$ according to the partitions $(\pi^n(O_1), \pi^n(O_2))$. An edge that ends in a state $\pi^n(p)$ in $\mathcal{A}$ is duplicated in two edges in the split graph, one going to $\pi^n(p)_1$, the other one going to $\pi^n(p)_2$. This split automaton $\mathcal{A}'$ can also be obtained by several consecutive elementary output state splittings. It admits a k-approximate eigenvector $\mathbf{v}'$ such that that $v'_{q_1} + v'_{q_2} = v_q$ for $q \in \Pi(p)$ and $v'_q = v_q$ otherwise. Moreover ϕ induces a permutation π' of the states of $\mathcal{A}'$ defined by $\pi'(q) = \pi(q)$ if $q \notin \Pi(p)$ and $\pi'(q_1) = \pi(q)_1$, $\pi'(q_2) = \pi(q)_2$ if $q \in \Pi(p)$. For any states $q, q' \in \Pi(p)$, $v'_{q_1} = v'_{q'_1}$ and $v'_{q_2} = v'_{q'_2}$. Thus for any state q of $\mathcal{A}'$, $v'_q = v'_{\pi'(q)}$.

The process can be iterated until one gets an automaton with a k-approximate eigenvector whose all components are equal. Its minimum outdegree is thus at least k. The final automaton is local, recognizes the same code, and ϕ induces a permutation π of its states. In each orbit of the permutation of the states, we choose one state q and a set O_q of k outgoing edges of q. Thus $\pi^n(O_q)$ is a set of k outgoing edges of $\pi^n(q)$. The automaton obtained by discarding the edges that have not been chosen, is a local automaton with a constant outdegree k. It recognizes a ϕ-invariant shift of finite type $T \subset S$ and ϕ induces the permutation π on its states. Let us denote by $\mathcal{B}$ this automaton.

Let $\mathcal{T}$ be a transducer whose output automaton is $\mathcal{B}$. We define the input labels of the edges of $\mathcal{T}$ as follows. In each orbit of the permutation π of the states, we choose one state q we label each of the k edges in O_q with a distinct letter of the alphabet A. For each state $\pi^n(q)$, we then define the input label of an edge e in $O_{\pi^n(q)}$ as the input tag of the edge $\pi^{-n}(e)$. Note that this possible since π has no fixed point. Since the output automaton of the transducer is local, it realizes a sliding block map d from T onto $A^{\mathbb{Z}}$. The map associates to a bi-infinite sequence $\mathbf{u}$ in T the input label of the unique bi-infinite path whose output label is $\mathbf{u}$. Since the input automaton of the transducer is deterministic, the map d is right-resolving. By construction, $d \circ \phi = d$ on T. Thus the transducer is ϕ-invariant. $\square$

By using codeterministic automata in Lemma 3.4, it is possible to construct in Theorem 3.3 a ϕ-invariant transducer which has a deterministic input automaton recognizing $A^{\mathbb{Z}}$, and a local output automaton recognizing $T \subset S$ which is moreover codeterministic with a finite delay.

We give below an example of the construction of Lemma 3.4 for the channel S of Figure 3.2 with $\pi = (ac)(b)$ on $B = \{a, b, c\}$. The capacity of this channel is $\log(2)$. We have $P_1^2 = \{ab, cb\}$, $P_2^2 = \{ac, ba, cc\}$, and $P_3^2 = \{aa, bc, ca\}$. After splitting states 2 and 3 one time each, we get the

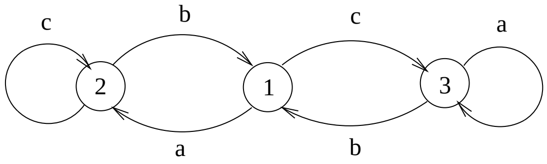

FIG. 3.2. *A channel S invariant by permutation*

automaton of Figure 3.3, where the set P_q^2 is given inside the state q.

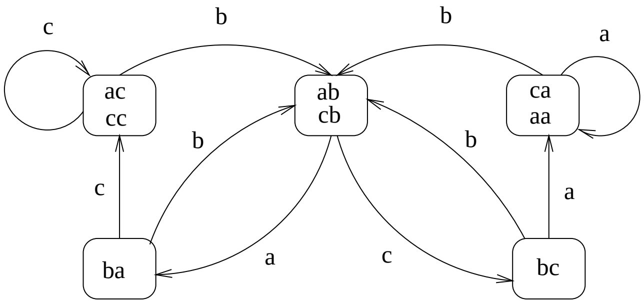

FIG. 3.3. *After two splittings*

After several rounds of state splittings we get the local automaton which is the output automaton of the transducer represented in Figure 3.4. A possible input tag such that the tranducer is ϕ-invariant is given in the same Figure. The input alphabet is the 2-letter alphabet $A = \{0, 1\}$. For

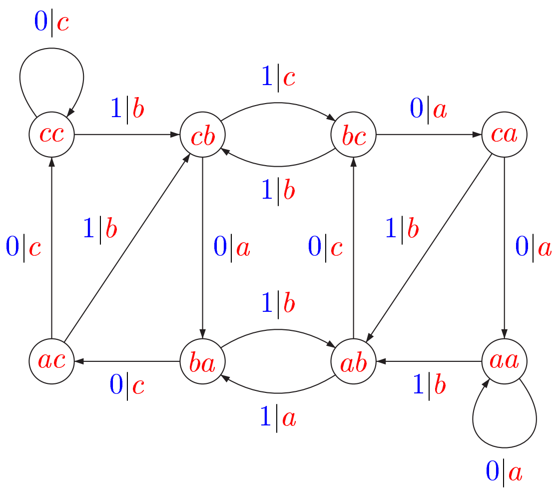

FIG. 3.4. *The coding/decoding transducer*

this choice of the input labeling, the decoding d is a 3-sliding block map and can thus be performed with a window of length 3. The sliding block window decoding property ensures that the decoder is immune to catastrophic error propagation.

Note that the hypothesis of Theorem 3.3 that the permutative conjugacy ϕ does not fix any any bi-infinite codeword is necessary. Indeed, let us consider a ϕ-invariant transducer with a deterministic input automaton and a (m, a)-local output automaton. By the sharpness condition, for any state q, $\phi(E_q) \cap E_q = \emptyset$, where E_q is the set of labels of bi-infinite paths going through q at the index 0. This implies that ϕ has no fixed point.

3.5. Extension to ϕ-invariant sofic channels. Coding schemes that allow to encode arbitrary sequences into a constrained system of sequences which is modelled by a sofic system which is not of finite type, have been obtained in [1], [20], [8, 7]. In order to get a sliding block decoder, and since we consider here transducer with a constant rate $1:1$, the capacity of the channel has to be strictly greater than the capacity of the source, or some other conditions on the channel have to be added, like to be *almost of finite type* [20]. In the former case, the corresponding coding theorem due to Marcus [24] (see also [1]) is the following.

THEOREM 3.5. *Let A and B be two finite alphabets. Let $S \subset B^{\mathbb{Z}}$ be a sofic system such that $\mathrm{cap}(S) > \mathrm{cap}(A^{\mathbb{Z}})$. Then there exist a subshift of finite type $T \subset S$ and a right-resolving and onto sliding block map d from T to $A^{\mathbb{Z}}$.*

In this section, we extend this result to ϕ-invariant sofic channels, i. e.

to sofic channels invariant by a permutative conjugacy ϕ. Since we consider channels whose capacity is strictly greater than the source capacity, this allows us to remove the hypothesis that ϕ has no fixed point. Recall from Section 3 that this hypothesis was necessary in the case of equality of the capacities of the source and of the channel. The construction that we use in the proof does not use the state splitting process of [1] but the method of poles for sofic channels presented in [8].

THEOREM 3.6. *Let A and B be two finite alphabets. Let $S \subset B^{\mathbb{Z}}$ be a sofic system such that* $\mathrm{cap}(S) > \mathrm{cap}(A^{\mathbb{Z}})$ *and ϕ be a permutative conjugacy of S. Then there exist a subshift of finite type $T \subset S$ and a right-resolving and onto sliding block map d from T to $A^{\mathbb{Z}}$ such that $d \circ \phi = d$.*

We shall use in the proof the following definitions.

We define the *rank* of a finite word u in a deterministic automaton as number of states q that end a path labeled by u. Recall that a synchronizing word is a word of rank 1. An automaton is said to be *follower-separated* if any two distinct states p, q have a different future, that is, there is a finite word u that is the label of some path starting at p and that is not the label of any path starting at q (or conversely). For any sofic shift S, there is a deterministic follower-separated automaton that recognizes it. It is moreover possible to choose an *essential* automaton, that is, an automaton such that each state has at least one outgoing edge end at least one incoming edge. If S is irreducible, it is its minimal automaton. It is known that, in such an automaton, any finite word can be extended on the right to a synchronizing word [23, Proposition 3.3.16].

We call *orbit* of a finite word u the finite set $\{\phi^n(u), n \geq 0\}$.

LEMMA 3.7. *Let $\mathcal{A}$ be an essential deterministic follower-separated automaton recognizing a ϕ-invariant sofic S, where ϕ is a permutative conjugacy which is not the identity. Then the automaton $\mathcal{A}$ admits a synchronizing word x such that $\phi(x) \neq x$ and each word in the orbit of x is a synchronizing word.*

Proof. If ϕ is not the identity, there is a bi-infinite word which is not constant by ϕ. By compacity there is a finite word u which is not constant by ϕ. We extend u on the right to a synchronizing word x which is not constant by ϕ. Since ϕ is permutative, there is a positive integer r such that ϕ^r is the identity. Let us assume that $x, \phi(x), \ldots \phi^i(x)$ are synchronizing, for $0 \leq i < r$. If $\phi^{(i+1)}(x)$ is not synchronizing, we extend it on the right to a synchronizing word $\phi^{(i+1)}(x)v$. Let $y = x\phi^{-(i+1)}(v)$. Then $y, \phi(y), \ldots \phi^{(i+1)}(y)$ are synchronizing. By induction, we get then that there is word which is not constant by ϕ and whose orbit contains only synchronizing words. □

Proof. [of Theorem 3.6] Let S be a ϕ-invariant sofic shift with $\mathrm{cap}(S) > \log(k)$. We can also assume that ϕ is not the identity map. By Lemma 3.7 there is an essential deterministic follower-separated automaton recognizing S such that there is synchronizing word x, non constant by ϕ, whose orbit contains only synchronizing words. Let m be the length of x. As in the proof of Lemma 3.4, we do input state splittings to get a deterministic automaton $\mathcal{A}$ recognizing S, such that for each state q of $\mathcal{A}$, P_q^m is reduced to one word. The word x and all its orbit are synchronizing words of $\mathcal{A}$. Let F be the set of states focused to by the orbit O of x. The states of F are $(m, 0)$-strongly synchronizing and ϕ permutes the states of F with a permutation, still denoted by π, and defined as follows. Let $q \in F$, if q is the state focused to by $u \in O$, then $\pi(q)$ is the state focused to by $\pi(u)$. Note that, since S is not of finite type, the automaton $\mathcal{A}$ is not local and some states may not be strongly synchronizing. The states of F are thus both strongly synchronizing and sharp.

Let z be a finite path, we denote by $l(z)$ its length. We recall the definition of the notion of principal states introduced by Franaszek in [16]. Let M be a positive integer. The set of *principal states*, relatively to k and M, is the maximal subset P of Q such that the following condition, called the capacity condition, is satisfied. For any state q in P there exist a prefix-free set Z_q of paths of length bounded by M, starting at q, terminating in P, and whose length distribution satisfies the Kraft equality for the integer k, i. e. $\sum_{z \in Z_q} k^{-l(z)} = 1$. A set of paths Z_q that satisfies these conditions is called *admissible*. An algorithm to compute P is given in [16]. It has been shown that if $\mathrm{cap}(S) > \log(k)$, then there is a positive integer M such that P is non empty. We define here a set of principal states, relatively to k and M, as the maximal subset P of the set F such that the capacity condition holds for P. It can be shown like in [8] that P is still non empty and can be computed with the same algorithm. If p is a principal state, P contains the whole orbit of p by π. We choose a representative state in each orbit of P.

For each representative state p, let Z_p be an admissible set of paths and z a path in Z_p labeled by u from p to another principal state q. Since p, q, $\pi(p)$, $\pi(q)$ are strongly synchronizing states, there is at least one path labeled by $\phi(u)$ going from $\pi(p)$ to $\pi(q)$. Since the automaton $\mathcal{A}$ is deterministic, and thus unambiguous, there it at most one path labeled by $\phi(u)$ going from $\pi(p)$ to $\pi(q)$. We denote this path by $\pi(z)$. Thus the sets $\pi^n(Z_p)$, where $n \geq 0$ and $\pi^n(Z_p) = \{\pi^n(z) \mid z \in Z_p\}$, are admissible. For any $n \geq 0$ we define the set $Z_{\pi^n(p)} = \pi^n(Z_p)$. We moreover choose an admissible set Z_p that satisfies an optimization condition of the kind described in [8] like minimizing, among all possible admissibles sets, the sum of the lengths of the words of the set. The computation of such a set Z_p and other possible optimization conditions are described in [8]. In order to decrease the complexity of the final coding transducer, it is better to have as many principal states as possible.

Now a transducer $\mathcal{T}$, that we shall use to encode and decode, is constructed as follows. Let p be a representive state of an orbit of P and Z_p be an admissible set of paths which satisfies an optimization condition. We choose a prefix-free set of words X_p on the k-letter alphabet A which has the same length distribution as Z_p. This is possible since this length distribution satisfies the Kraft equality for the integer k. We choose a length-preserving bijection ρ_p from Z_p to X_p. We define then, for any $n \geq 0$, the set $X_{\pi^n(p)}$ by $X_{\pi^n(p)} = X_p$, and the length-preserving bijection $\rho_{\pi^n(p)}$ from $Z_{\pi(p)}$ to $X_{\pi(p)}$ by $\rho_{\pi^n(p)}(z) = \rho(\pi^{-n}(z))$. Note that this is possible only since $\pi(p) \neq p$.

For any principal state p we defined a state $\hat{p}$ of $\mathcal{T}$, called a *pole*. For any principal state p and each path z in Z_p going from p to a principal state q, one defines a path $\hat{z}$ in $\mathcal{T}$ of length $l(z)$ from $\hat{p}$ to $\hat{q}$ with $l(z) - 1$ dummy states of $\mathcal{T}$ strung along the path $\hat{z}$. The input label $\rho(z)$ is assigned to $\hat{z}$ while its output label is the label of the path z in $\mathcal{A}$.

It is proved in [8] that the optimization condition on the sets Z_q guarantee that the above constructed transducer has a local output automaton. It recognizes a shift of finite type $T \subset S$. Since the sets X_q are prefix-free codes that satisfy the Kraft equality for the integer k, the input automaton of $\mathcal{T}$ is deterministic with a finite delay and recognizes $A^{\mathbb{Z}}$. Finally, by construction, the decoding is ϕ-invariant. The sliding block decoding map d is thus an onto right closing map from T to $A^{\mathbb{Z}}$ such that $d \circ \phi = d$. A standard transformation of the transducer allows us to get a transducer with the same property and with a deterministic input [21]. ☐

We briefly discuss some possible extensions of Theorems 3.3 and 3.6. Since in [20], Theorem 3.2 has been extended to the class of almost of finite type shifts, we conjecture that Theorem 3.3 can be extended to this case. Another open question is the extension of Theorem 3.3 to all sofic shifts, in the case of equality of capacities between the source and the channel, with a decoding map which is not a sliding block map but does not propagate errors (see [20]), and is invariant by ϕ. Beside these possible extensions, it should be remarked that the hypothesis on the conjugacy ϕ of being permutative is very strong. It seems nevertheless difficult to weaken this hypothesis to obtain similar results for more general conjugacies. Finally, it has been proved by Ashley in [3] (see also [2]) that sliding block decoders with a linear window size can be constructed for constrained channels. The same theoretical complexity problem can be considered for ϕ-invariant decoders.

REFERENCES

[1] R. L. Adler, D. Coppersmith, and M. Hassner, *Algorithms for sliding block codes*, IEEE Trans. Inform. Theory, IT-29 (1983), pp. 5–22.

[2] J. J. Ashley, *A linear bound for sliding block decoder window size*, IEEE Trans. Inform. Theory, 34 (1988), pp. 389–399.

[3] ———, *A linear bound for sliding block decoder window size (II)*, IEEE Trans. Inform. Theory, 42 (1996), pp. 1913–1924.

[4] J. J. Ashley and M.-P. Béal, *A note on the method of poles for code construction*, IEEE Trans. Inform. Theory, 40 (1994), pp. 512–517.

[5] M.-P. Béal, *Codes circulaires, automates locaux et entropie*, Theoret. Comput. Sci., 57 (1988), pp. 288–302.

[6] ———, *The method of poles : a coding method for constrained channels*, IEEE Trans. Inform. Theory, 36 (1990), pp. 763–772.

[7] M.-P. Béal, *Codage Symbolique*, Masson, 1993.

[8] M.-P. Béal, *Extensions of the method of poles for code construction*, Tech. Rep. 97–26, I.G.M., Université de Marne-la-Vallée, 1997.

[9] M.-P. Béal and D. Perrin, *Symbolic dynamics and finite automata*, in Handbook of Formal Languages, G. Rozenberg and A. Salomaa, eds., vol. 2, Springer-Verlag, 1997, ch. 10.

[10] S. Benedetto, R. Garello, M. Mondin, and M. D. Trott, *Rotational invariance of trellis codes. part II: Group codes and decoders*, IEEE Trans. Inform. Theory, 42 (1996), pp. 766–778.

[11] M. Boyle, B. P. Kitchens, and B. H. Marcus, *A note on minimal covers for sofic systems*, Proc. AMS, 95 (1985), pp. 403–411.

[12] A. R. Calderbank, *The differential encoding of coset codes by algebraic methods*, in Coding Theory and Design Theory, Part I: Coding Theory, D. Ray-Chaudhuri, ed., New York: Springer-Verlag, 1990, pp. 16–34.

[13] H. J. Chizeck and M. D. Trott, *Algebraic systems, trellis codes, and rotational invariance*, in Advances in Electronics and Electron Physics, vol. 79, New York: Academic Press, 1990, pp. 1–71.

[14] G. D. Forney Jr., *Coset codes-Part I: Introduction and geometrical classification*, IEEE Trans. Inform. Theory, 34 (1988), pp. 1123–1151.

[15] G. D. Forney Jr., B. H. Marcus, N. T. Sindhushayana, and M. D. Trott, *Multilingual dictionary: System theory, coding theory, symbolic dynamics and automata theory*, in Different Aspects of Coding Theory, R. Calderbank, ed., vol. 50, AMS Proc. Symp. Appl. Math, 1995, pp. 109–138.

[16] P. A. Franaszek, *On synchronous variable length coding for discrete noiseless channels*, Inform. Control, 1-J (1969), pp. 155–164.

[17] ———, *A general method for channel coding*, IBM J. Res. Dev., 24 (1980), pp. 638–641.

[18] ———, *Coding for constrained channel: a comparison of two approaches*, IBM J. Res. Dev., 33 (1989), pp. 602–608.

[19] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, *Codes for digital recorders*, IEEE Trans. Inform. Theory, IT-44 (1998), pp. 2260–2299.

[20] R. Karabed and B. H. Marcus, *Sliding block coding for input-restricted channels*, IEEE Trans. Inform. Theory, 34 (1988), pp. 2–26.

[21] B. P. Kitchens, *Continuity properties of factor maps in ergodic theory*, Ph.D. thesis, University of North Carolina, Chapel Hill, 1981.

[22] ———, *Symbolic Dynamics: one-sided, two-sided and countable state Markov shifts*, Springer-Verlag, 1997.

[23] D. A. Lind and B. H. Marcus, *An Introduction to Symbolic Dynamics and Coding*, Cambridge, 1995.

[24] B. H. Marcus, *Sofic systems and encoding data*, IEEE Trans. Inform. Theory, IT-31 (1985), pp. 366–377.

[25] B. H. Marcus, R. M. Roth, and P. H. Siegel, *Constrained systems and coding for recording channels*, in Handbook of Coding Theory, V. Pless and W. Huffman, eds., vol. II, North Holland, 1998, ch. 20, pp. 1635–1764.

[26] B. H. Marcus, P. H. Siegel, and J. K. Wolf, *Finite-state modulation codes for date storage*, IEEE J. Sel. Areas Commun., 10 (1992), pp. 5–37.

[27] M. Oerder, *Rotationally invariant trellis codes for m-PSK modulation*, IEEE Int. Conf. on Communications. Conf. Rec., (1985), pp. 552–556.

[28] J. G. Proakis, *Digital Communications*, Mac Graw Hill, 1995.

[29] D. Tang and L. Bahl, *Block codes for a class of constrained noiseless channels*, Inform. Contr., 17 (1970), pp. 436–461.

[30] M. D. Trott, S. Benedetto, R. Garello, and M. Mondin, *Rotational invariance of trellis codes. part I: Encoders and precoders*, IEEE Trans. Inform. Theory, 42 (1996), pp. 751–765.

[31] L. F. Wei, *Rotationally invariant convolutional channel coding with expanded signal space. part I:* 180°, IEEE J. Sel. Areas Commun., SAC-2 (1984), pp. 659–671.

[32] ———, *Rotationally invariant convolutional channel coding with expanded signal space. part II: Nonlinear codes*, IEEE J. Sel. Areas Commun., SAC-2 (1984), pp. 672–686.