Aperiodic Two-way Transducers and FO-Transductions

Olivier Carton[1] and Luc Dartois[2,3]

1   **LIAFA, Université Paris Diderot, France**
    Olivier.Carton@liafa.univ-paris-diderot.fr
2   **LIF, UMR7279 Aix-Marseille Université & CNRS, France**
    luc.dartois@lif.univ-mrs.fr
3   **Centrale Marseille, France**

─── **Abstract** ───

Deterministic two-way transducers on finite words have been shown by Engelfriet and Hoogeboom to have the same expressive power as **MSO**-transductions. We introduce a notion of aperiodicity for these transducers and we show that aperiodic transducers correspond exactly to **FO**-transductions. This lifts to transducers the classical equivalence for languages between **FO**-definability, recognition by aperiodic monoids and acceptance by counter-free automata.

1   Introduction

The regularity of a language of finite words is a central notion in theoretical computer science. Combining several seminal results, it is equivalent whether a language is

(a) accepted by a (non-)deterministic one-way or two-way automaton [22] and [27],

(b) described by a regular expression [17],

(c) defined in (Existential) Monadic Second Order (**MSO**) logic [8],

(d) the preimage by a morphism into a finite monoid [20].

Since then, the characterization of fragments of **MSO** has been a very successful story. Using this equivalence between different formalisms, several fragments of **MSO** have been characterized by algebraic means and shown to be decidable. Combining results of Schützenberger [25] and of McNaughton and Papert [19] yields, for instance, that a language of finite words is First Order (**FO**) definable if and only if all the groups contained in its syntactic monoid are trivial (aperiodic). From the results of Schützenberger [26] and others [28], it is also known that a language is First Order definable with two variables (**FO**^{2}) if and only if its syntactic monoid belongs to the class **DA** which is easily decidable.

Automata can be equipped with output to make them compute functions and relations. They are then called transducers. Note then that all variants are no longer equivalent as they are as acceptors. Deterministic transducers compute a subclass of rational functions called sequential functions [9]. Two-way transducers are also more powerful than one-way transducers (see Example 1). The study of transducers has many applications. Transducers are used to model coding schemes (compression schemes, convolutional coding schemes, coding schemes for constrained channels, for instance). They are also widely used in computer arithmetic [15], natural language processing [24] and programs analysis [11].

The equivalence between automata and **MSO** has been first lifted to transducers and the functions they realize by Engelfriet and Hoogeboom [13]. They show that a function from words to words can be realized by a deterministic two-way transducer if and only it is a **MSO**-transduction. First, this result deals surprisingly with two-way transducers rather than one-way transducers which are much simpler. Second, the **MSO**-definability used for automata is replaced by **MSO** graphs transductions defined by Courcelle [12]. A **MSO**-transduction is a function where the output graph is defined as a **MSO**-interpretation into a fixed number of copies of the input graph. In the result of Engelfriet and Hoogeboom, words are seen as linear graphs whose vertices carry the symbols.

1.1 Contribution

In this paper, we combine the approach of Engelfriet and Hoogeboom with the one of Schützenberger, McNaughton and Papert. We introduce a notion of aperiodicity for two-way transducers and we show that it corresponds to **FO**-transductions. By **FO**-transduction, we mean **MSO**-transduction where the interpretation is done through **FO**-formulas. The definition of aperiodicity is achieved by associating a transition monoid with each two-way transducer. The construction of this algebraic object is already implicit in the literature [27, 21, 6]. In order to obtain our result, we have considered a different logical signature for transductions from the one used in [13]. In [13], the signature contains the symbol predicates to check symbols carried by vertices and the edge predicate of the graph. Since words are viewed as linear graphs, this is the same as the signature with the successor relation on words. In our result, the signature contains the symbol predicates and the order (of the linear graph). This is equivalent for **MSO**-transductions since the order can easily be defined with the successor by a **MSO**-formula. This is however not equivalent any more for **FO**-transductions that we consider. With this signature, the definition of **FO**-transduction requires that the order on the output word can be defined by a **FO**-formula. The change in the signature is necessary to obtain the result.

1.2 Related work

The aperiodic rational functions, that is, functions realized by a one-way transducer with an aperiodic transition monoid have already been characterized in [23]. This characterization is not based on logic but rather on the inverse images of aperiodic languages.
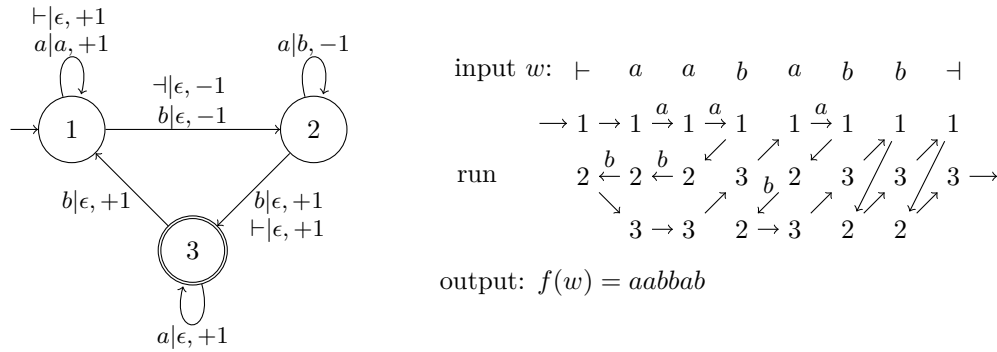
The notion of aperiodic two-way transducer was already defined and studied in [18], although their model defined length-preserving functions and the transducers had both their reading and writing heads moving two-way. The assumption that the function is length preserving makes the relation between the input and the output easier to handle.

Recently, Bojanczyk, in [7], also characterized first-order definable transducers for machines using a finer but more demanding semantic, the so-called origin semantic.

In [3], Alur and Černý defined the streaming string transducers, a one-way deterministic model equivalent to deterministic two-way transducers and **MSO** transductions. More recently, Filiot, Krishna and Trivedi proposed in [14] a definition of transition monoid for this model. They also proved that aperiodic and 1-bounded streaming string transducers have the same expressive power as **FO** transductions, which is one of the models considered by our main result.

1.3 Structure

The paper is organized as follows. Definitions of two-way transducers and FO-transductions are provided in Section 2. The construction of the transition monoid associated with a

Figure 1 A transducer and its run over $w = aababb$.

transducer is given there. The main result is stated in Section 3. Section 4 focuses on one aspect of the stability by composition of functions realized by aperiodic two-way transducers. It is one of the main ingredients used in the proof of the main result. The proof itself is sketched in Sections 5 and 6.

2   Definitions

In this section, we present the different models that will be used throughout the article.

2.1   Two-way transducers

A transducer is an automaton equipped with outputs. While an input word is processed along a run by the transducer, each used transition outputs some word. All these output words are concatenated to form the output of the run. The automaton might be one-way or two-way but we mainly consider two-way transducers in this paper. When the transducer is non-deterministic, there might be several runs and therefore several output words for a single input word. All two-way transducers considered in this paper are deterministic. For each input word, there is then at most one valid run and one output word. The partial function which maps each input word to the corresponding output word is said to be *realized* by the transducer. The automaton obtained by forgetting the outputs is called the *input automaton* of the transducer.

A two-way transducer is a very restricted variant of a Turing machine with an input and an output tape. First, the input tape is read-only. Second, the output tape is write-only and the head on this tape only moves forwards. Written symbols on this tape cannot be over-written later by other symbols.

▶ **Example 1.** Let A be the alphabet $\{a, b\}$. Let us consider, as a running example, the function $f : A^* \to A^*$ which maps each word $w = a^{k_0}ba^{k_1} \cdots ba^{k_n}$ to the word $f(w) = a^{k_0}b^{k_0}a^{k_1}b^{k_1} \cdots a^{k_n}b^{k_n}$ obtained by adding after each block of consecutive a a block of consecutive b of the same length. Since each word w over A can be uniquely written $w = a^{k_0}ba^{k_1} \cdots ba^{k_n}$ with some k_i being possibly equal to zero, the function f is well defined. The word $w = aababb = a^2ba^1ba^0ba^0$ is mapped to $f(w) = a^2b^2a^1b^1a^0b^0a^0b^0 = aabbab$.

This function is realized by the transducer depicted in Figure 1. This transducer proceeds as follows to compute $f(w)$ from the input word w. While being in state 1 and moving forwards, it copies a block of consecutive a to the output. While in state 2 and moving backwards, the corresponding block of b is written to the output. While being in state 3, the

transducer moves forwards writing nothing until it reaches the next block of consecutive a. Note that this function cannot be realized by a one-way transducer.

Formally, a two-way transducer is defined as follows:

▶ **Definition 2** (Two-way transducer). A (*deterministic*) *two-way transducer* $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, A, B, \delta, \gamma, q_0, F))$ defined as follows:

- Q is a finite *state set*.
- A and B are the *input* and *output alphabet*.
- $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \to Q \times \{-1, 0, +1\}$ is the *transition function*. Contrary to the one-way machines, the transition function also outputs an integer, corresponding to the move of the reading head. The alphabet is enriched with two new symbols $\vdash$ and $\dashv$, which are endmarkers that are added respectively at the beginning and the end of the input word, such that for all $q \in Q$, we have $\delta(q, \vdash) \in Q \times \{0, +1\}$ and $\delta(q, \dashv) \in Q \times \{-1, 0\}$.
- $\gamma : Q \times (A \uplus \{\vdash, \dashv\}) \to B^*$ is the *production function*.
- $q_0 \in Q$ is the *initial state*.
- $F \subseteq Q$ is the set of final states.

The transducer $\mathcal{A}$ processes finite words over A. If at state p the symbol a is processed and $\delta(p, a) = (q, d)$, then $\mathcal{A}$ moves to state q, moves the reading head to the left or right depending on d, and outputs $\gamma(p, a)$.

Let $w = a_1 \cdots a_n$ be a fixed finite word over A and $a_0 = \vdash$ and $a_{n+1} = \dashv$. Whenever $\delta(p, a_m) = (q, d)$ and $\gamma(p, a_m) = v$, we write $(p, m) \xrightarrow{|v} (q, n)$ where $n = m + d$. We do not write the input over the arrow because it is always the symbol below the reading head, namely, a_m. In this notation, the pairs represent the current configuration of a machine with the current state and the current position of the input head. A *run* of the transducer over w is a finite sequence of consecutive transitions

$$(p_0, m_0) \xrightarrow{|v_1} (p_1, m_1) \cdots (p_{n-1}, m_{n-1}) \xrightarrow{|v_n} (p_n, m_n)$$

and we write $(p_0, m_0) \xrightarrow{|v} (p_n, m_n)$ where $v = v_1 v_2 \cdots v_n$. We also refer to finite runs over words w when all positions m_i in the run but the last are between 1 and $|w|$. The last position m_n is allowed to be between 0 and $|w| + 1$. It is 0 if the run leaves w on the left end and it is $|w| + 1$ if it leaves $|w|$ on the right end.

A run $(p_0, m_0) \xrightarrow{|v} (p_n, m_n)$ over a marked word $\vdash u \dashv$ is *accepting* if it starts at the first position in the initial state and ends on the right endmarker $\dashv$ in a final state. Then v is the *image* of u by $\mathcal{A}$, denoted $\mathcal{A}(u) = v$.

2.2 Transition monoid

In order to define a notion of aperiodicity for a transducer, we associate with each two-way automaton a monoid called its *transition monoid*. A transducer is then called *aperiodic* if the transition monoid of its input automaton is aperiodic. Let us recall that a monoid is called aperiodic if it contains no trivial group [2]. Equivalently, a monoid M is aperiodic if there exists a smallest integer n, called the *aperiodicity index*, such that for any element x of M, we have $x^n = x^{n+1}$. Note first that the transition monoid of a transducer is the transition monoid of its input automaton and does not depend of its outputs. Note also that our definition is sound for either deterministic or non-deterministic automata/transducers although we only use it for deterministic ones. Lastly, remark that it extends naturally the notion of transition monoid for one-way automata.
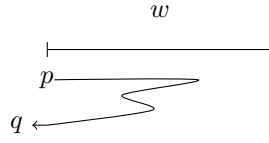
Figure 2 A left-to-left behavior (p, q) of a word w.

The transition monoid is, as usual, obtained by quotienting the free monoid A^* by a congruence which captures the fact that two words have the same *behavior* in the automaton. In an one-way automaton $\mathcal{A}$, the behavior of a word w is the set of pairs (p, q) of states such that there exists a run from p to q in $\mathcal{A}$. Two words are then considered equivalent if their respective behaviors contain the same pairs of states. In a two-way automaton, the behavior of a word is also characterized by the runs it contains but since the reading head can move both ways, the behavior is split into four behaviors called left-to-left, left-to-right, right-to-left and right-to-right behaviors. We only define the left-to-left behavior $\mathrm{bh}_{\ell\ell}(w)$ of a word w. The three other behaviors $\mathrm{bh}_{\ell r}(w)$, $\mathrm{bh}_{r\ell}(w)$ and $\mathrm{bh}_{rr}(w)$ are defined analogously.

Let $\mathcal{A}$ be a two-way automaton. The *left-to-left behavior* $\mathrm{bh}_{\ell\ell}(w)$ of w in $\mathcal{A}$ is the set of pairs (p, q) such that there exists a run which starts at the first position of w in state p and leaves w on the left end in state q (see Figure 2).

Before defining the transition monoid, we illustrate the notion of behavior on the transducer depicted in Figure 1.

▶ **Example 3.** Consider the transducer depicted in Figure 1 and the word $w = aab$. From the run depicted in Figure 1, it can be inferred that

$$\mathrm{bh}_{\ell\ell}(w) = \{(1, 2), (2, 2)\} \qquad \mathrm{bh}_{r\ell}(w) = \{(1, 2)\}$$
$$\mathrm{bh}_{\ell r}(w) = \{(3, 1)\} \qquad \mathrm{bh}_{rr}(w) = \{(2, 3), (3, 1)\}.$$

▶ **Definition 4** (Transition monoid). Let $\mathcal{A} = (Q, A, \delta, q_0, F)$ be a two-way automaton. The transition monoid of $\mathcal{A}$ is $A^*/\sim_{\mathcal{A}}$ where $\sim_{\mathcal{A}}$ is the conjunction of the four relations $\sim_{\ell\ell}$, $\sim_{\ell r}$, $\sim_{r\ell}$ and $\sim_{rr}$ defined for any words w, w' of A^* as follows :

- $w \sim_{\ell\ell} w'$ if $\mathrm{bh}_{\ell\ell}(w) = \mathrm{bh}_{\ell\ell}(w')$.
- $w \sim_{\ell r} w'$ if $\mathrm{bh}_{\ell r}(w) = \mathrm{bh}_{\ell r}(w')$.
- $w \sim_{r\ell} w'$ if $\mathrm{bh}_{r\ell}(w) = \mathrm{bh}_{r\ell}(w')$.
- $w \sim_{rr} w'$ if $\mathrm{bh}_{rr}(w) = \mathrm{bh}_{rr}(w')$.

The neutral element of this monoid is the class of the empty word ϵ, whose behaviors $bh_{xy}(\epsilon)$ is the identity function if $x \neq y$, and is the empty relation otherwise.

These relations are not new and were already evoked in [21, 6] for example. Moreover, the left-to-left behavior was already introduced in [27] to prove the equivalence between one-way and two-way automata.

For a deterministic two-way automaton, the four behaviors $\mathrm{bh}_{\ell\ell}(w)$, $\mathrm{bh}_{\ell r}(w)$, $\mathrm{bh}_{r\ell}(w)$ and $\mathrm{bh}_{rr}(w)$ are partial functions. In the non-deterministic case, these four relations are not functions but relations over the state set Q because there might exist several runs with the same starting state and different ending states. Furthermore, for deterministic automaton, the domains of the functions $\mathrm{bh}_{\ell\ell}(w)$ and $\mathrm{bh}_{\ell r}(w)$ (resp. $\mathrm{bh}_{r\ell}(w)$ and $\mathrm{bh}_{rr}(w)$) are disjoint, since there is a unique run starting in state p at the first (resp. last) position of w. Thus a run starting at the first (resp. last) position leaves w either on the left or the right. For a deterministic two-way automaton, the four behaviors $\mathrm{bh}_{\ell\ell}(w)$, $\mathrm{bh}_{\ell r}(w)$, $\mathrm{bh}_{r\ell}(w)$ and $\mathrm{bh}_{rr}(w)$

$$[a] = a^+$$
$$[b] = b$$
$$[ab] = a^+b$$
$$[ba] = ba^+$$
$$[aba] = aA^*bA^*a$$
$$[abb] = aA^*bA^*b$$
$$[bba] = bA^*bA^*a$$
$$[bb] = bA^*b$$

^{*1}	

$^*a$	b

ab	ba

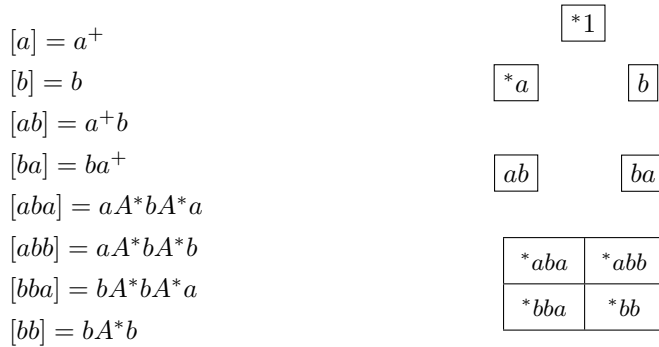|	
$^*aba$	$^*abb$
$^*bba$	$^*bb$

■ **Figure 3** The equivalence classes of the transition monoid and its $\mathcal{D}$-class representation

can be seen as a single partial function f_w from $Q \times \{\ell, r\}$ to $Q \times \{\ell, r\}$ where $f_w(p, x) = (q, y)$ whenever $(p, q) \in \mathrm{bh}_{xy}(w)$ for any $x, y \in \{\ell, r\}$.

▶ **Lemma 5.** *Let $\mathcal{A}$ be a two-way transducer. Then the relation $\sim_{\mathcal{A}}$ is a congruence of finite index.*

It is pure routine to check that $\sim_{\mathcal{A}}$ is indeed a congruence. It is of finite index since each of the four relations $\sim_{\ell\ell}$, $\sim_{\ell r}$, $\sim_{r\ell}$ and $\sim_{rr}$ has at most $2^{|Q|^2}$ classes. Note that the composition of the behaviors is not as straightforward as in the case of one-way automata, the four relations being intertwined. For example, the composition law of the $\mathrm{bh}_{\ell r}$ relation is given by the equality $\mathrm{bh}_{\ell r}(uv) = \mathrm{bh}_{\ell r}(u)\big(\mathrm{bh}_{\ell\ell}(v)\,\mathrm{bh}_{rr}(u)\big)^* \mathrm{bh}_{\ell r}(v)$ which follows from the decomposition of a run in uv.

▶ **Example 6.** We illustrate the notion of a transition monoid by giving the one of the transducer depicted in Figure 1. We have omitted all words containing one of the two endmarkers since these words cannot contribute to a group. The eight classes of the congruence $\sim_{\mathcal{A}}$ for the remaining words are given in Figure 3 on the left. The $\mathcal{D}$-class representation of this monoid is also given for the aware reader on the right. It can be checked that this monoid is aperiodic. The transducer of Figure 1 is then aperiodic.

2.3 FO graph transductions

The **MSO**-transductions defined by Courcelle [12] are a variant of the classical logical interpretation of a relational structure into another one. Let us recall that a relational structure S has a $\mathcal{L}$-interpretation, for some logic $\mathcal{L}$, into a structure T if it has an isomorphic copy in T defined by $\mathcal{L}$-formulas. More precisely, this means that there exists a $\mathcal{L}$-formula φ_S with one first-order free variable and a one-to-one correspondence f between the domain of S and the subset T' of elements of T satisfying φ_S. Furthermore, for each relation R of S with arity r, there exists a $\mathcal{L}$-formula φ_R with r first-order free variables such that R is isomorphic via f to the r-tuples of T' satisfying φ_R.

A **MSO**-transduction defines for each input structure a new structure obtained by **MSO**-interpretation into a fixed number of copies of the input structure. In this case, the relations are the letter predicates and the successor relation, which are of arity one and two respectively. To fit into this framework, words are viewed as linear graphs. Each word $w = a_1 \cdots a_n$ is viewed as a linear graph with n vertices carrying the symbols $a_1, \ldots, a_n$. Linear means here that if the vertex set is $\{1, 2, \ldots, n\}$, the edge set is $\{(k, k+1) : 1 \le k \le n-1\}$.
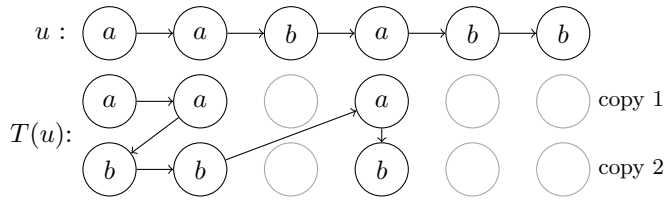
Figure 4 The linear graph of $u = aababb$ and the output structure of T over u.

When restricted to linear graphs, the **MSO**-transductions has been proved to have the same expressive power as two-way transducers [13]. We are interested in this article in **FO** graph transductions, the restriction to first order formulas. Since we consider transductions whose domain is not the set of all graphs, there is an additional closed formula φ_{dom} which determines whether the given graph is in the domain of the transduction.

Before giving the formal definition, we give below an example of a **FO**-transduction. Note that when considering **FO** transductions, the successor relation is replaced by the order relation.

▶ **Example 7.** We give here a **FO** graph transduction that realizes the function f introduced in Example 1. So let $T = (A, A, \varphi_{dom}, C, \varphi_{pos}, \varphi_{\leqslant})$ be the **FO** graph transduction defined as follows :

- $A = \{a, b\}$ is both the input and output alphabet,
- $C = \{1, 2\}$,
- φ_{dom} is a **FO** formula stating that the input is a linear graph,
- $\varphi_a^1(x) = \varphi_b^2(x) = \mathbf{a}(x)$, the other position formulas being set as $false$,
- the order formulas are defined now :
  - $\varphi_{\leqslant}^{i,i}(x, y) = x \leqslant y$ for $i = 1, 2$,
  - $\varphi_{\leqslant}^{1,2}(x, y) = x \leqslant y \vee (\forall z\ y \leqslant z \leqslant x \rightarrow \mathbf{a}(z))$,
  - $\varphi_{\leqslant}^{2,1}(x, y) = \exists z\ x \leqslant z \leqslant y \wedge \mathbf{b}(z)$.

▶ **Definition 8.** A **FO**-graph transduction is a tuple $T = (A, B, \varphi_{dom}, C, \varphi_{pos}, \varphi_{\leqslant})$ defined as follows:

- A is the *input alphabet*.
- B is the *output alphabet*.
- φ_{dom} is the *domain formula*. A graph is accepted as input if it satisfies the domain formula.
- C is a finite set, denoting the copies of the input that can exist in the output.
- φ_{pos} is a set of formulas with one free variable $\varphi_b^c(x)$, for $b \in B$ and $c \in C$. Given c, the formulas $\varphi_b^c(x)$, for $b \in B$, are mutually exclusive. The c copy of a node i is labelled by b if, and only if, the formula $\varphi_b^c(x/i)$ is true.
- $\varphi_{\leqslant}$ is a set of formulas with two free variables $\varphi_{\leqslant}^{c,c'}(x, y)$, for $c, c' \in C$. There exists a path from the c copy of a node i to the c' copy of a node j if, and only if, the formula $\varphi_{\leqslant}^{c,c'}(x/i, y/j)$ is true.

All formulas are required to be in **FO**[$<$] and are evaluated on the input graph.

The output graph is defined as a substructure of the C copies of the input linear graph, in which a node exists if it satisfies one position formula, and is labelled accordingly, and the order is defined according to the order formulas.

In this article, we are only interested in linear graph transductions, which only accept words seen as linear graphs as input. An input word has an *image* by a **FO** graph transduction

if the associated linear graph satisfies its domain formula and the order relation of the output graph, defined by the order formulas, defines a linear graph corresponding to a word. If one condition fails, then the function is undefined on the given input. One should note that the fact that a graph is linear and corresponds to a word is **FO**-definable.

In Figure 4, we give the output structure of T over the linear graph $u = aababb$. Note that for the sake of readability, we do not draw the whole order relation, but simply the successor relation.

3   Main result

We are now ready to state the main result of this article, as an extension of the result by McNaughton and Papert [19] and Schützenberger [25] in the context of two-way transducers and **MSO** transductions established by Engelfriet and Hoogeboom [13].

▶ **Theorem 9.** *The functions realized by aperiodic two-way transducers are exactly the functions realized by* **FO** *graph transductions over words.*

The theorem is proved in Sections 5 and 6. The first inclusion relies on Theorem 13, while the second inclusion stems from the conjunction of Theorems 18, 19 and 20. The next Section is devoted to the composition of transducers, which is a key tool of the proof.

4   Composition of transducers

As transducers realize functions over words, the natural question of the compositionality occurs. In a generic way, this question is : given two functions realized by some machine, can we construct a machine that realizes the composition of these functions. This question has been considered in [16] for generic machines, and resolved positively in the case of deterministic two-way transducers in [10].

This result can also be obtained using the equivalence of two-way transducers with **MSO** transductions, since these are easily proved to be stable by composition (see [12]). However, the reduction from **MSO** transductions to two-way transducers established in [13] makes an extensive use of a weaker version of this result, which is that the composition of a one-way deterministic, called sequential in the following, transducer with a two-way transducer can be done by a two-way transducer, which was first proved in [1].

In this section, we follow this approach, and now prove that this result holds for aperiodic transducers, in the sense that if the two input transducers are aperiodic, then we can construct an aperiodic transducer realizing the composition.

▶ **Theorem 10.** *Let $\mathcal{A}$ be a sequential transducer that can be composed with a two-way transducer $\mathcal{B}$, both deterministic and aperiodic. Then we can effectively construct an aperiodic and deterministic two-way transducer $\mathcal{C}$ such that $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$.*

5   From aperiodic two-way transducers to FO transductions

Let us consider a deterministic and aperiodic two-way transducer. We aim to construct a first-order graph transduction that realizes the same function.

In order to do that, we need to define a formula φ_{dom} for the input domain, formulas φ_{pos} for each copies of a position and each output letter of $\mathcal{A}$, and, contrary to the generic case of **MSO** graph transductions where only the successor is defined, we need here to define

order formulas $\varphi_{\leqslant}$ that describe the order relation on the output depending on the copies of the nodes from the input.

The following result simply stems from the equivalence of aperiodic monoids and first order logic established in [25, 19], but is an essential step to link aperiodicity to first-order, as it is used in the next theorem, which proves that the order relation between positions is first-order definable.

▶ **Lemma 11.** *Let $\mathcal{A} = (Q, A, \delta)$ be an aperiodic two-way automaton. Then the relation classes of $\sim_{\ell\ell}$, $\sim_{\ell r}$, $\sim_{r\ell}$, $\sim_{rr}$ and consequently $\sim_{\mathcal{A}}$ of $\mathcal{A}$ are* **FO**-*definable.*

▶ **Lemma 12.** *Let $\mathcal{A}$ be an aperiodic two-way automaton. Then for any pair of states q and q' of $\mathcal{A}$, there exists a* **FO**-*formula $\varphi^{q,q'}(x, y)$ such that for any word u in the domain of $\mathcal{A}$ and any pair of positions i and j of u,*

$$u \models \varphi^{q,q'}(x/i, y/j)$$

if, and only if, the run of $\mathcal{A}$ over u starting at position i in state q eventually reaches the position j in state q'.

We now state the main result of this section and construct the first-order transduction that realizes $\mathcal{A}$.

▶ **Theorem 13.** *Let $\mathcal{A}$ be an aperiodic two-way transducer. Then we can effectively construct a* **FO**-*graph transduction that realizes the same function as $\mathcal{A}$.*

Proof. For simplicity of the proof, we consider a transducer $\mathcal{A} = (Q, A, B, \delta, \gamma, i, F)$ where the production of any transition is at most one letter. This can be done without loss of generality, since any given transducer can be normalized this way by increasing the number of states. We now give the formal definition of the **FO** transduction $T = (A, B, \varphi_{dom}, Q, \varphi_{pos}, \varphi_{\leqslant})$ that realizes $\mathcal{A}$.

As we consider string transductions within the scope of graph transductions, the domain formula also has to ensure that the input is a linear graph. This can be done in **FO** by a formula stating that there is one position that has no predecessor, one position that has no successor, every other position has exactly one successor and one predecessor and every pair of positions is comparable. Then the domain formula of T is the formula describing the language recognized by the input automaton of $\mathcal{A}$ conjuncted with the linear graph formula. By Lemma 11, as $\mathcal{A}$ is aperiodic the domain formula is **FO**-definable. The order formulas are given by Lemma 12, where obviously $\varphi_{\leqslant}^{q,q'}(x, y) = \varphi^{q,q'}(x, y)$.

The $\varphi_b^q(x)$ formulas, where $q \in Q$ and $b \in B$, express that the production of $\mathcal{A}$ at the position quantified by x in state q is b, but also that the run of $\mathcal{A}$ over u reaches the said position in state q. Should we define $A_{b,q} = \{a \in A \mid \gamma(a, q) = b\}$, then the first condition is expressed as $\bigvee_{a \in A_{b,q}} a(x)$. The second condition is then equivalent to saying that there exists a run from the initial state of $\mathcal{A}$ to the current position, which is expressed by the formula $\exists y \forall z \ y \leqslant z \wedge \varphi^{i,q}(y, x)$. The formula $\varphi_b^q(x)$ is thus defined as the conjunction of these two formulas.

The transduction T is now defined. All formulas are expressed in the first order logic, and it realizes the same function as $\mathcal{A}$, proving the theorem.   ◀

6   From **FO** transductions to aperiodic two-way transducers

The proof scheme for this inclusion is adapted from the one in [13] proving that **MSO** transductions are realized by two-way deterministic transducers. We prove that we can

construct an aperiodic two-way transducer with **FO** look around from a **FO** transduction, and that the constructions given in [13] suppressing the look around part preserve the aperiodicity.

We define in the next subsection the models of transducers with look-around that are used in the proof. We then give an alternative definition of aperiodicity which can be applied to transducers with logic look-around before explaining the constructions that lead up to the result.

6.1 Transducers with look-around

Here, we define two kinds of transducers with look-around. The first one is a restriction of two-way transducers with regular look-around, where we limit the regular languages used in the tests to Star-free languages, which is the rational characterization of first-order logic. These transducers differ from the classic ones by their transitions, where the tests are not determined by the letter read, but also by the prefix and suffix which can be evaluated according to some regular languages.

The second extension we consider is transducers with first-order look around. In this case, the selection of a transition, as well as the movements of the reading head, are determined by formulas. Formal definitions are given below.

In both cases only the definition of transition is changed, the definition of run and accepting run remaining the same.

▶ **Definition 14** (two-way transducer with Star-Free look around). *Two-way transducers with Star-Free look around* are a subclass of two-way transducers with regular look around defined in [13], where all languages in the tests are Star-Free.

Formally, it is a machine $\mathcal{A} = (Q, A, B, \Delta, i, F)$ where Q, A, B, i and F are the same as for two-way transducers, and transitions and productions are regrouped in Δ, and are of the form (q, t, q', v, m) where q and q' are states from Q, $v \in B^*$ is the production of the transition, $m \in \{-1, 0, +1\}$ describes the movement of the reading head and t is a test of the form (L_p, a, L_s) where a is a letter of $A \uplus \{\vdash, \dashv\}$, and L_p and L_s are Star-Free languages over the same alphabet. A test (L_p, a, L_s) is satisfied if the reading head is on a position labelled by the letter a, the prefix of the input word up to the position of the reading head belongs to L_p, and symmetrically the suffix belongs to L_s.

Such a machine is deterministic if the tests performed in a given state are mutually exclusives.

▶ **Definition 15** (two-way transducer with **FO** look around). *Two-way transducers with **FO** look around* are a subclass of two-way transducers with **MSO** look around where formulas are restricted to the first-order.

Formally, it is a machine $\mathcal{A} = (Q, A, B, \Delta, i, F)$ where Q, A, B, i and F are the same as two-way transducers, and transitions of Δ are of the form $(q, \varphi(x), q', v, \psi(x, y))$ where q and q' are states from Q, $v \in B^*$ is the production of the transition and $\varphi(x)$ and $\psi(x, y)$ are **FO** formulas with respectively one and two free variables. A transition $(q, \varphi(x), q', v, \psi(x, y))$ can be taken if the formula $\varphi(x)$ holds on the input word, where x quantifies the current position i of the reading head, say $\vdash u \dashv \models \varphi(x/i)$. Then the reading head moves to a position j such that $\vdash u \dashv \models \psi(x/i, y/j)$.

Such a machine is deterministic if the unary tests appearing in a given state are mutually exclusive, and if for any input word u, any movement formula $\psi(x, y)$ and any position i, there exists at most one position j such that $\vdash u \dashv \models \psi(x/i, y/j)$.

6.2 Aperiodicity by path contexts

The reading head of transducers with logic look-around can jump several positions at a time and in any direction. Then the notion of behavior for such transducers becomes blurry, since behaviors would have to be considered starting at any position, and moreover the direction taken while exiting a word is not decided locally, but depends on the context.

We thus give an equivalent characterization of the aperiodicity of a transducer through all contexts at a time, for machines whose reading head does not move position by position.

We recall that given a (deterministic) transducer $\mathcal{A} = (Q, A, B, \Delta, init, F)$ and u an input word of $\mathcal{A}$, the accepting path of $\mathcal{A}$ over u, denoted $path(u)$, is the sequence $(q_0, i_0) \ldots (q_n, i_n)$ of pairs from $Q \times [0, |u| + 1]$ (the length of u plus the endmarkers) describing the behavior of the reading head of $\mathcal{A}$ while reading u, as defined in Subsection 2.1.

We now define the projection of paths, as a way to highlight some information and forget the rest. It is applied to contexts in order to only retain the influence of a word on its context.

▶ **Definition 16** (Projection and context paths)**.** Let $I = [i_1, \ldots, i_k]$ be an ordered sequence of integers. We define $path_I(u)$ as the sequence of pairs from $Q \times \{1, \ldots, k\}$ such that for any pairs (q, j) and (q', j'), (q, j) appears before (q', j') in $path_I(u)$ if, and only if, (q, i_j) appears before $(q', i_{j'})$ in $path(u)$. Informally, this corresponds to selecting pairs whose position is in I and renaming them according to the set I.

Abusing notations, we will note the *context path* $path_{vw}(vuw) = path_I(vuw)$ where I is the set of positions of v and w.

Then $path_{vw}(vuw)$ is the trace of the run over u on the context v, w and two words u and u' are $\mathcal{A}$-equivalent if for any context v, w, we have equality of the paths contexts $path_{vw}(vuw) = path_{vw}(vu'w)$. Then by definition of the aperiodicity, a transducer or an automaton $\mathcal{A}$ is *aperiodic* if there exists a positive integer n such that for any words u, v and w on the input alphabet of $\mathcal{A}$, the context paths $path_{vw}(vu^n w)$ and $path_{vw}(vu^{n+1} w)$ are equals. One should remark that on two-way transducers, this notion is equivalent to the aperiodicity of the transition monoid. The next lemma serves as the link from the first-order logic to the aperiodicity by context paths.

▶ **Lemma 17.** *Let T be a **FO** graph transduction. There exists a positive integer n such that for any input words u, v and w such that $vu^n w$ is in the domain of T, $vu^{n+1} w$ is also in the domain of T and the two words satisfy the same formulas of T, when the free variables quantify positions of v or w.*

Proof. First consider the domain formula of T. Since it is a **FO** formula, it has an aperiodicity index n, in the sense that for any words u, v and w, $vu^n w$ is in the domain of T if, and only if, $vu^{n+1} w$ is in the domain of T.

We now prove the result in the case where i ranges over v and j ranges over w, but similar proofs hold for i and j ranging independently over v and w. Consider a pair c, c' of copies in C, and integers $0 \leqslant i < |v|$ and $0 \leqslant j < |w|$. Then a word with positions i and j quantified respectively by x and y can be seen as a word over the alphabet $A \times \{0, 1\}^2$, where all letters have $(0, 0)$ as second component, except $(v_i, 1, 0)$ and $(w_j, 0, 1)$. The formula $\varphi_{\leqslant}^{c,c'}(x, y)$ can then be equivalently seen as a closed formula over this enriched alphabet. This formula being in **FO**, it describes an aperiodic language, and then there exists an integer n' such that $vu^{n'} w$ satisfies $\varphi_{\leqslant}^{c,c'}(x/i, y/|vu^{n'}| + j)$ if, and only if, $vu^{n'+1} w$ satisfies $\varphi_{\leqslant}^{c,c'}(x/i, y/|vu^{n'+1}| + j)$.

A similar argument also holds for the node formulas $\varphi_b^c(x)$. As there is a finite number of formulas, there exists an integer, the maximum of the index of each formulas, such that the result holds. ◀

This lemma means that the transducer has an aperiodicity index, in the sense that u^n and u^{n+1} behave the same way for the same context. It also corresponds to the notion of aperiodicity defined earlier in this section, where the sequence ranges over pairs of copy and position.

6.3   Construction of the aperiodic transducer

We now hold all the necessary tools to prove the reduction from **FO** transductions to aperiodic two-way transducers.

We present the first construction, from a **FO** transduction to a two-way transducer with **FO** look around. The construction is quite simple. By putting the copy set of the graph transduction as the set of states of the transducer, we can use the fact that the reading head of a transducer with logic look around jumps between positions to strictly follow the output structure of the input transduction. We then use Lemma 17 to prove the aperiodicity of the construction.

▶ **Theorem 18.** *Let T be a **FO** graph transduction. Then we can effectively construct an aperiodic two-way transducer with **FO** look around that realizes the same function over words.*

We have now constructed an aperiodic two-way machine from the input **FO** transduction. But even though two-way transducers with **MSO** look around are known to be equivalent to two-way transducers [13], we need to prove that we can suppress the **FO** look around while preserving the aperiodicity of the construction. This is done by the two following theorems, using Star-free look around as an intermediate step. We show that the construction evoked in [13] do preserve the aperiodicity, leading to the result.

▶ **Theorem 19.** *Given an aperiodic two-way transducers with **FO** look around, we can construct an aperiodic two-way transducers with Star-Free look around that realizes the same function.*

Proof. In order to prove this theorem, we rely on the proof of Lemma 6 from [13], which proves that two-way transducers with **MSO** look around can be expressed by two-way transducers with regular look around. This is done by constructing a transducer whose regular tests stem directly from the **MSO** formulas. Then the reading head simulates the jumps of the reading head of the transducer with **MSO** look around by moving step by step up to the required position.

We aim to prove on one hand that if the formulas are defined in the first-order, then the resulting two-way transducer only uses Star-free look around, and on the other hand that if moreover the input transducer is aperiodic, then the output transducer is also aperiodic.

The first claim is proved by noticing that the languages used in the regular look around construction are languages defined by formulas of the input transducer with **FO** look around with free variables. Then if the free variables are seen as an enrichment of the alphabet, similarly to what is done in the proof of Lemma 17, the formula remains first-order, and consequently all the languages used in look around tests are Star-free.

Now let us compare the moves of the reading head of the resulting transducer with Star-free look around with the ones of the head of the input transducer with **FO** look around. The path of the resulting transducer over any word can entirely be deduced from the path of the input transducer, by adding step by step walks between the jumps of the reading head. Then, if the input transducer is aperiodic with index n, given three words u, v and w, the context paths $path_{vw}(vu^n w)$ and $path_{vw}(vu^{n+1} w)$ are equal, and thus the context paths for

the transducer with Star-free look around, that are deduced from it, are equal too, proving the aperiodicity of the resulting transducer.   ◄

We finally prove that we can suppress the Star-free look around tests while preserving the aperiodicity, which concludes the proof of the main theorem.

▶ **Theorem 20.** *Given an aperiodic two-way with Star-free look around, we can construct an aperiodic two-way transducers that realizes the same function.*

Proof. Here again, we consider the construction from [13], Lemma 4, proving that we can suppress the regular look around tests. Our goal is then to prove that this operation preserves the aperiodicity when the input transducer only uses Star-free languages.

The construction relies heavily on the fact that the composition of a two-way transducer with a one-way transducer can be done by a two-way transducer. It is used to preprocess the input by adding the result of each regular tests from the transitions at each position. Given the test (L_p, a, L_s) of a transition, a left-to-right pass simulates L_p and reproduces the input where each position is enriched with the information : does its prefix belong to L_p. Symmetrically, a right-to-left transducer adds the same information for L_s. Let us remark that since these languages are Star-free, the input automaton of the transducers simulating these languages are aperiodic.

Then the information regarding every transition is added to the input, and lastly we can construct a two-way transducer that acts in the same way as the input transducer, but where all the look around have been suppressed and are done locally by looking at the enrichment part of the letter. This two-way transducer without look around is aperiodic if the input transducer is aperiodic, since they share the same paths, and thus context paths.

Finally, the input transducer is given as the composition of a single aperiodic two-way transducer with a finite number of aperiodic one-way transducers. Should we first remark that, by symmetry of the problem, Theorem 10 also holds for right sequential transducers, through several uses of this composition result we finally obtain a unique two-way transducer that realizes the input transducer with Star-free look around.   ◄

7   Conclusion

We recall that a similar work has been done for streaming string transducers by Filiot, Krishna and Trivedi [14]. Then through **FO** transductions, this result and Theorem 9 prove the equivalence of aperiodicity for the two models of transducers.

There exists algorithms that input a two-way transducer and construct directly an equivalent streaming string transducer (see [4] for example). It would be interesting to check first if the aperiodicity is preserved through these algorithms, and secondly to compare the size and aperiodicity indexes of the two transition monoids. Although unknown from the authors, a reciprocal procedure and its study would hold the same interest.

On a more generic note, one can ask which fragments of logic preserve their algebraic characterization in the scope of two-way transducers and **MSO** transductions. For example, are $\mathcal{J}$-trivial transducers equivalent to $\mathcal{B}\Sigma_1$ transductions ? The main challenges for this question are the stability by composition of these restricted classes of transducers on one hand, and on the other hand the very definition of logic transductions for restricted fragments, as a fragment must retain some fundamental expressive properties, such as being able to characterize linear graphs.

Finally, we would like to point out the fact that even if we can decide if a given two-way transducer is aperiodic, it is still open to decide if the function realized by a two-way

transducer can be realized by an aperiodic one. An promising approach for this problem might be to consider machine-independent descriptions of functions, as defined recently for streaming string transducers in [5] for example. This was successfully done in [7] for machines with origin semantic. We also think that this question could be solved by the notion of canonical object of a function over words, which has yet to be defined.

References

1   A. V. Aho, J. E. Hopcroft, and J. D. Ullman. A general theory of translation. *Math. Systems Theory*, 3:193–221, 1969.

2   Jorge Almeida. *Finite semigroups and universal algebra*, volume 3 of *Series in Algebra*. World Scientific Publishing Co., Inc., River Edge, NJ, 1994. Translated from the 1992 Portuguese original and revised by the author.

3   Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *30th International Conference on Foundations of Software Technology and Theoretical Computer Science (STACS'10)*, volume 8 of *LIPIcs – Leibniz International Proceedings in Informatics*, pages 1–12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, 2010.

4   Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 65–74. IEEE Computer Society, 2012.

5   Rajeev Alur, Adam Freilich, and Mukund Raghothaman. Regular combinators for string transformations. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14–18, 2014*, page 9. ACM, 2014.

6   Jean-Camille Birget. Concatenation of inputs in a two-way automaton. *Theoret. Comput. Sci.*, 63(2):141–156, 1989.

7   Mikolaj Bojanczyk. Transducers with origin information. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8–11, 2014, Proceedings, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.

8   J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.

9   Christian Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theoret. Comput. Sci.*, 5:325–338, 1977.

10 Michal P. Chytil and Vojtěch Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *Automata, languages and programming (Fourth Colloq., Univ. Turku, Turku, 1977)*, pages 135–137. Lecture Notes in Comput. Sci., Vol. 52. Springer, Berlin, 1977.

11 A. Cohen and J.-F. Collard. Instance-wise reaching definition analysis for recursive programs using context-free transductions. In *PACT'98*, 1998.

12 Bruno Courcelle. Monadic second-order definable graph transductions: a survey [see MR1251992 (94f:68009)]. *Theoret. Comput. Sci.*, 126(1):53–75, 1994. Seventeenth Colloquium on Trees in Algebra and Programming (CAAP'92) and European Symposium on Programming (ESOP) (Rennes, 1992).

13   Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.

14   Emmanuel Filiot, Shankara Narayanan Krishna, and Ashutosh Trivedi. First-order definable string transformations. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS'14, December 15–17, 2014, New Delhi, India*, volume 29 of *LIPIcs – Leibniz International Proceedings in Informatics*, pages 147–159. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, 2014.

15   Christiane Frougny. Numeration systems. In M. Lothaire, editor, *Algebraic Combinatorics on Words.* Cambridge, 1999. to appear.

16   J. E. Hopcroft and J. D. Ullman. An approach to a unified theory of automata. *Bell System Tech. J.*, 46:1793–1829, 1967.

17   S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata studies*, Annals of mathematics studies, no. 34, pages 3–41. Princeton University Press, Princeton, N. J., 1956.

18   Pierre McKenzie, Thomas Schwentick, Denis Thérien, and Heribert Vollmer. The many faces of a translation. In *Automata, languages and programming (Geneva, 2000)*, volume 1853 of *Lecture Notes in Comput. Sci.*, pages 890–901. Springer, Berlin, 2000.

19   Robert McNaughton and Seymour Papert. *Counter-free automata.* The M.I.T. Press, Cambridge, Mass.-London, 1971.

20   A. Nerode. Linear automaton transformation. In *Proceeding of the AMS*, volume 9, pages 541–544, 1958.

21   J.-P. Pécuchet. Automates boustrophédon, semi-groupe de Birget et monoïde inversif libre. *RAIRO Inform. Théor.*, 19(1):71–100, 1985.

22   M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3, 1959.

23   Christophe Reutenauer and Marcel-Paul Schützenberger. Variétés et fonctions rationnelles. *Theoretical Computer Science*, 145:229–240, 1995.

24   Emmanuel Roche and Yves Schabes. *Finite-State Language Processing*, chapter 7. MIT Press, Cambridge, 1997.

25   Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

26   Marcel-Paul Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13(1):47–75, 1976/77.

27   J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Develop.*, 3:198–200, 1959.

28   Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation. In *ACM STOCS'98*, pages 256–263, 1998.