

# FO-definable transformations of infinite strings

Vrunda Dave<sup>1</sup>, Shankara Narayanan Krishna <sup>\*1</sup>, and Ashutosh Trivedi <sup>†1,2</sup>

1 Indian Institute of Technology Bombay ([krishnas, vrunda@cse.iitb.ac.in](mailto:krishnas, vrunda@cse.iitb.ac.in))

2 University of Colorado Boulder ([ashutosh.trivedi@colorado.edu](mailto:ashutosh.trivedi@colorado.edu))

---

## Abstract

The theory of regular and aperiodic transformations of finite strings has recently received a lot of interest. These classes can be equivalently defined using logic (Monadic second-order logic and first-order logic), two-way machines (regular two-way and aperiodic two-way transducers), and one-way register machines (regular streaming string and aperiodic streaming string transducers). These classes are known to be closed under operations such as sequential composition and regular (star-free) choice; and problems such as functional equivalence and type checking, are decidable for these classes. On the other hand, for infinite strings these results are only known for regular transformations: Alur, Filiot, and Trivedi studied transformations of infinite strings and introduced an extension of streaming string transducers over infinite strings and showed that they capture monadic second-order definable transformations for infinite strings. In this paper we extend their work to recover connection for infinite strings among first-order logic definable transformations, aperiodic two-way transducers, and aperiodic streaming string transducers.

1998 ACM Subject Classification F.1.1

Keywords and phrases Transducers, FO-definability, Infinite Strings

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.

## 1 Introduction

The beautiful theory of regular languages is the cornerstone of theoretical computer science and formal language theory. The perfect harmony among the languages of finite words definable using abstract machines (deterministic finite automata, nondeterministic finite automata, and two-way automata), algebra (regular expressions and finite monoids), and logic (monadic second-order logic (MSO) [7]) set the stage for the generalizations of the theory to not only for the theory of regular languages of infinite words [8, 17], trees [4], partial orders [22], but more recently for the theory of regular transformations of the finite strings [6], infinite strings [3, 1], and trees [2]. For the theory of regular transformations it has been shown that abstract machines (two-way transducers [13] and streaming string transducers [6]) precisely capture the transformations definable via monadic second-order logic transformations [10]. For a detailed exposition on the regular theory of languages and transformations, we refer to the surveys by Thomas [22, 23] and Filiot [14], respectively.

There is an equally appealing and rich theory for first-order logic (FO) definable subclasses of regular languages. McNaughton and Papert [18] observed the equivalence between FO-definability and star-free regular expressions for finite words, while Ladner [16] and Thomas [24] extended this connection to infinite words. The equivalence of star-free regular

---

\* Partly supported by CEFIPRA project AVERTS.

† Supported by research sponsored by DARPA under agreement number FA8750-15-2-0096. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.



© Dave, Krishna, and Trivedi;  
licensed under Creative Commons License CC-BY

36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016).

Editors: Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen; Article No. ; pp. 1–14



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

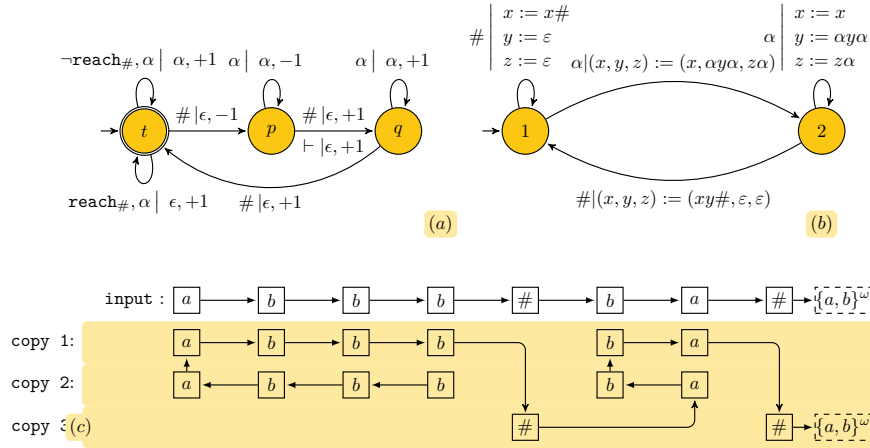


Figure 1 Transformation  $f_1$  given as (a) two-way transducers with look-ahead (b) streaming string transducers with  $F(\{2\}) = xz$  is the output associated with Muller set  $\{2\}$ , and (c) FO-definable transformation for the string  $abbb\#ba\#\{a, b\}^\omega$ . Here symbol  $\alpha$  stands for both symbols  $a$  and  $b$ , and the predicate  $\text{reach}_\#$  is the lookahead that checks whether string contains a  $\#$  in future.

expressions and languages defined via aperiodic monoids is due to Schützenberger [20] and corresponding extension to infinite words is due to Perrin [19]. For a detailed introduction to FO-definable language we refer the reader to Diekert and Gastin [12].

The results for the theory of FO-definable transformations are relatively recent. While Courcelle’s definition of logic based transformations [10] provides a natural basis for FO-definable transformations of finite as well as infinite words, [15] observed that over finite words, streaming string transducers [6] with an appropriate notion of aperiodicity precisely capture the same class of transformations. Carton and Dartois [9] introduced aperiodic two-way transducers for finite words and showed that it precisely captures the notion of FO-definability. We consider transformations of infinite strings and generalize these results by showing that appropriate aperiodic restrictions on two-way transducers and streaming string transducers on infinite strings capture the essence of FO-definable transformations. Let us study an example to see how the following  $\omega$ -transformation can be represented using logic, two-way transducers, and streaming string transducers.

► **Example 1 (Example Transformation).** Let  $\Sigma = \{a, b, \#\}$ . Consider an  $\omega$ -transformation  $f_1 : \Sigma^\omega \rightarrow \Sigma^\omega$  such that it replaces any maximal  $\#$ -free finite string  $u$  by  $\bar{u}u$ , where  $\bar{u}$  is the reverse of  $u$ . Moreover  $f_1$  is defined only for strings with finitely many  $\#$ ’s, e.g. for all  $w = u_1\#u_2\#\dots\#u_n\#v$  s.t  $u_i \in \{a, b\}^*$  and  $v \in \{a, b\}^\omega$ , we have  $f_1(w) = \bar{u}_1u_1\#\dots\#\bar{u}_nu_n\#v$ .

**Logic based transformations.** Logical descriptions of transformations of structures—as introduced by Courcelle [10]—work by introducing a fixed number of copies of the vertices of the input graph; and the domain, the labels and the edges of the output graph are defined by MSO formulae with zero, one or two free variables, respectively, interpreted over the input graph. Figure 1(c) shows a way to express transformation  $f_1$  using three copies of the input with a) logical formula  $\phi_{\text{dom}}$  expressing the domain of the transformation, b) logical formulae  $\phi_\alpha^c(i)$  (with one free variable) for every copy  $c \in \{1, 2\}$  and letter  $\alpha \in \{a, b\}$  expressing the label of a position  $i$  for copy  $c$ , and c) logical formulae  $\phi^{c,d}(i, j)$  with two free variables expressing the edge from position  $i$  of copy  $c$  to position  $j$  of copy  $d$ . The formulae

$\phi_{\text{dom}}$ ,  $\phi_a^c$ , and  $\phi^{c,d}$  are interpreted over input structure (in this paper always an infinite string), and it is easy to see that these formulae for our example can easily be expressed in MSO. In this paper we study logical transformations expressible with FO and to cover a larger class of transformations, we use natural order relation  $<$  for positions instead of the successor relation. We will later show that the transformation  $f_1$  indeed can be expressed using FO.

**Two-Way Transducers.** For finite string transformations, Engelfriet and Hoogeboom [13] showed that the finite-state transducers when equipped with a two-way input tape have the same expressive power as MSO transducers, and Carton and Dartois [9] recovered this result for FO transducers and two-way transducers with aperiodicity restriction. A crucial property of two-way finite-state transducers exploited in these proofs [13, 9] is the fact that transitions capable of regular (star-free) *look-ahead* (i.e., transitions that test the whole input string against a regular property) do not increase the expressiveness of regular (aperiodic) two-way transducers. However, this property does not hold in case of  $\omega$ -strings. In Figure 1(a), we show a two-way transducer characterizing transformation  $f_1$ . The transducer uses the lookahead  $\text{reach}_{\#}$  to check if the remaining part of the string contains a  $\#$  in future. A transition labeled  $\langle \phi, \alpha | \beta, +1 \rangle$  of the two-way transducer should be read as: if the current position on the string satisfies the look-ahead  $\phi$  and the current symbol is  $\alpha$  then output symbol  $\beta$  and move the input tape head to the right. This transducer works by first checking if the string contains a  $\#$  in the future of the current position, if so it moves its head all the way to the position before  $\#$  and starts outputting the symbols in reverse, and when it sees the end-marker or a  $\#$  it prints the string before the  $\#$ ; however, if there is no  $\#$  in future, then the transducer outputs the rest of the string. It is straightforward to verify that this transducer characterizes the transformation  $f_1$ . However, in the absence of the look-ahead a two-way transducer can not express this transformation.

**Streaming String Transducers.** Alur and Černý [6, 5] proposed a one-way finite-state transducer model, called the *streaming string transducers* (SST), that manipulates a finite set of string variables to compute its output, and showed that they have same expressive power as MSO transducers. SST, instead of appending symbols to the output tape, concurrently update all string variables using a concatenation of string variables and output symbols in a *copyless* fashion, i.e. no variable occurs more than once in each concurrent variable update. The transformation of a string is then defined using an output (partial) function  $F$  that associates states with a copyless concatenation of string variables, s.t. if the state  $q$  is reached after reading the string and  $F(q)=xy$ , then the output string is the final valuation of  $x$  concatenated with that of  $y$ . [3] generalized this by introducing a Muller acceptance condition to give an SST to characterize  $\omega$ -transitions. Figure 1(b) shows a streaming string transducer accepting the transformation  $f_1$ . It uses three string variables and concurrently prepends and/or appends these variables in a copyless fashion to construct the output. The acceptance set and the output is characterized by a Muller set (here  $\{2\}$  and its output  $xz$ ), such that if the infinitely visiting states set is  $\{2\}$  then the output is limit of the values of the concatenation  $xz$ . Again, it is easy to verify that SST in Figure 1(b) captures the transformation  $f_1$ .

**Contributions and Challenges.** Our main contributions include the definition of aperiodic streaming string transducers and aperiodic two-way transducers, and the proof of the following key theorem connecting FO and transducers for transformations of infinite strings.

► **Theorem 2.** *Let  $F : \Sigma^\omega \rightarrow \Gamma^\omega$ . Then the following assertions are equivalent:*

1.  *$F$  is first-order definable.*
2.  *$F$  is definable by some aperiodic two-way transducer with star-free look-around.*
3.  *$F$  is definable by some aperiodic streaming string transducers.*

We introduce the notion of transition monoids for automata, 2WST, and SST with the Muller acceptance condition; and recover the classical result proving aperiodicity of a language using the aperiodicity of the transition monoid of its underlying automaton. The equivalence between FOT and 2WST with star-free look-around (Section 4), crucially uses the transition monoid with Muller acceptance, which is necessary to show aperiodicity of the underlying language of the 2WST. On the other hand, while going from aperiodic SST to FOT (Section 5), the main difficulty is the construction of the FOT using the aperiodicity of the SST, and while going from 2WST with star-free look-around to SST (Section 6), the hard part is to establish the aperiodicity of the SST. Due to space limitation, we only provide key definitions and sketches of our results—complete proofs and related supplementary material can be found in longer version of this paper [11].

## 2 Preliminaries

A finite (infinite) string over alphabet  $\Sigma$  is a finite (infinite) sequence of letters from  $\Sigma$ . We denote by  $\epsilon$  the empty string. We write  $\Sigma^*$  for the set of finite strings,  $\Sigma^\omega$  for the set of  $\omega$ -strings over  $\Sigma$ , and  $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$  for the set of finite and  $\omega$ -strings. A language  $L$  over an alphabet  $\Sigma$  is defined as a set of strings, i.e.  $L \subseteq \Sigma^\infty$ .

For a string  $s \in \Sigma^\infty$  we write  $|s|$  for its length; note that  $|s| = \infty$  for an  $\omega$ -string  $s$ . Let  $\text{dom}(s) = \{1, 2, 3, \dots\}$  be the set of positions in  $s$ . For all  $i \in \text{dom}(s)$  we write  $s[i]$  for the  $i$ -th letter of the string  $s$ . For two  $\omega$ -strings  $s, s' \in \Sigma^\omega$ , we define the distance  $d(s, s')$  as  $\frac{1}{2^j}$  where  $j = \min\{k \mid s[k] \neq s'[k]\}$ . We say that a string  $s \in \Sigma^\omega$  is the limit of a sequence  $s_1, s_2, \dots$  of  $\omega$ -strings  $s_i \in \Sigma^\omega$  if for every  $\epsilon > 0$ , there is an index  $n_\epsilon \in \mathbb{N}$  such that for all  $i \geq n_\epsilon$ , we have that  $d(s, s_i) \leq \epsilon$ . Such a limit, if exists, is unique and is denoted as  $s = \lim_{i \rightarrow \infty} s_i$ . For example,  $b^\omega = \lim_{i \rightarrow \infty} b^i c^i$ .

### 2.1 Aperiodic Monoids for $\omega$ -String Languages

A monoid  $\mathcal{M}$  is an algebraic structure  $(M, \cdot, e)$  with a non-empty set  $M$ , a binary operation  $\cdot$ , and an identity element  $e \in M$  such that for all  $x, y, z \in M$  we have that  $(x \cdot (y \cdot z)) = ((x \cdot y) \cdot z)$ , and  $x \cdot e = e \cdot x$  for all  $x \in M$ . We say that a monoid  $(M, \cdot, e)$  is *finite* if the set  $M$  is finite. A monoid that we will use in this paper is the *free* monoid,  $(\Sigma^*, \cdot, \epsilon)$ , which has a set of finite strings over some alphabet  $\Sigma$  with the empty string  $\epsilon$  as the identity.

We define the notion of acceptance of a language via monoids. A morphism (or homomorphism) between two monoids  $\mathcal{M} = (M, \cdot, e)$  and  $\mathcal{M}' = (M', \times, e')$  is a mapping  $h : M \rightarrow M'$  such that  $h(e) = e'$  and  $h(x \cdot y) = h(x) \times h(y)$ . Let  $h : \Sigma^* \rightarrow M$ , be a morphism from free monoid  $(\Sigma^*, \cdot, \epsilon)$  to a finite monoid  $(M, \cdot, e)$ . Two strings  $u, v \in \Sigma^*$  are said to be similar with respect to  $h$  denoted  $u \sim_h v$ , if for some  $n \in \mathbb{N} \cup \{\infty\}$ , we can factorize  $u, v$  as  $u = u_1 u_2 \dots u_n$  and  $v = v_1 v_2 \dots v_n$  with  $u_i, v_i \in \Sigma^+$  and  $h(u_i) = h(v_i)$  for all  $i$ . Two  $\omega$ -strings are  $h$ -similar if we can find factorizations  $u_1 u_2 \dots$  and  $v_1 v_2 \dots$  such that  $h(u_i) = h(v_i)$  for all  $i$ . Let  $\cong$  be the transitive closure of  $\sim_h$ .  $\cong$  is an equivalence relation. Note that since  $M$  is finite, the equivalence relation  $\cong$  is of finite index. For  $w \in \Sigma^\infty$  we define  $[w]_h$  as the set  $\{u \mid u \cong w\}$ . We say that a morphism  $h$  accepts a language  $L \subseteq \Sigma^\infty$  if  $w \in L$  implies  $[w]_h \subseteq L$  for all  $w \in \Sigma^\infty$ .

We say that a monoid  $(M, \cdot, e)$  is *aperiodic* [21] if there exists  $n \in \mathbb{N}$  such that for all  $x \in M$ ,  $x^n = x^{n+1}$ . Note that for finite monoids, it is equivalent to require that for all  $x \in M$ , there exists  $n \in \mathbb{N}$  such that  $x^n = x^{n+1}$ . A language  $L \subseteq \Sigma^\omega$  is said to be aperiodic iff it is recognized by some morphism to a finite and aperiodic monoid [11].

## 2.2 First-Order Logic for $\omega$ -String Languages

A string  $s \in \Sigma^\omega$  can be represented as a relational structure  $\Xi_s = (\text{dom}(s), \preceq^s, (L_a^s)_{a \in \Sigma})$ , called the string model of  $s$ , where  $\text{dom}(s) = \{1, 2, \dots\}$  is the set of positions in  $s$ ,  $\preceq^s$  is a binary relation over the positions in  $s$  characterizing the natural order, i.e.  $(x, y) \in \preceq^s$  iff  $x \leq y$ ;  $L_a^s$ , for all  $a \in \Sigma$ , are the unary predicates that hold for the positions in  $s$  labeled with the letter  $a$ , i.e.,  $L_a^s(i)$  iff  $s[i] = a$ , for all  $i \in \text{dom}(s)$ . When it is clear from context we will drop the superscript  $s$  from the relations  $\preceq^s$  and  $L_a^s$ .

Properties of string models over the alphabet  $\Sigma$  can be formalized by first-order logic denoted by  $\text{FO}(\Sigma)$ . Formulas of  $\text{FO}(\Sigma)$  are built up from variables  $x, y, \dots$  ranging over positions of string models along with *atomic formulae* of the form  $x=y$ ,  $x \preceq y$ , and  $L_a(x)$  for all  $a \in \Sigma$  where formula  $x=y$  states that variables  $x$  and  $y$  point to the same position, the formula  $x \preceq y$  states that position corresponding to variable  $x$  is not larger than that of  $y$ , and the formula  $L_a(x)$  states that position  $x$  has the label  $a \in \Sigma$ . Atomic formulae are connected with *propositional connectives*  $\neg, \wedge, \vee, \rightarrow$ , and *quantifiers*  $\forall$  and  $\exists$  that range over node variables and we use usual semantics for them. We say that a variable is *free* in a formula if it does not occur in the scope of some quantifier. A *sentence* is a formula with no free variables. We write  $\phi(x_1, x_2, \dots, x_k)$  to denote that at most the variables  $x_1, \dots, x_k$  occur free in  $\phi$ . For a string  $s \in \Sigma^*$  and for positions  $n_1, n_2, \dots, n_k \in \text{dom}(s)$  we say that  $s$  with valuation  $\nu = (n_1, n_2, \dots, n_k)$  satisfies the formula  $\phi(x_1, x_2, \dots, x_k)$  and we write  $(s, \nu) \models \phi(x_1, x_2, \dots, x_k)$  or  $s \models \phi(n_1, n_2, \dots, n_k)$  if formula  $\phi$  with  $n_i$  as the interpretation of  $x_i$  is satisfied in the string model  $\Xi_s$ . The language defined by an FO sentence  $\phi$  is  $L(\phi) \stackrel{\text{def}}{=} \{s \in \Sigma^\omega : \Xi_s \models \phi\}$ . We say that a language  $L$  is FO-definable if there is an FO sentence  $\phi$  such that  $L = L(\phi)$ . The following is a well known result.

► **Theorem 3.** [18][20] *A language  $L \subseteq \Sigma^*$  is FO-definable iff it is aperiodic.*

## 2.3 Aperiodic Muller Automata for $\omega$ -String Languages

A deterministic Muller automaton (DMA) is a tuple  $\mathcal{A} = (Q, q_0, \Sigma, \delta, F)$  where  $Q$  is a finite set of states,  $q_0 \in Q$  is the initial state,  $\Sigma$  is an input alphabet,  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function, and  $F \subseteq 2^Q$  are the accepting (Muller) sets. For states  $q, q' \in Q$  and letter  $a \in \Sigma$  we say that  $(q, a, q')$  is a transition of the automaton  $\mathcal{A}$  if  $\delta(q, a) = q'$  and we write  $q \xrightarrow{a} q'$ . We say that there is a run of  $\mathcal{A}$  over a finite string  $s = a_1 a_2 \dots a_n \in \Sigma^*$  from state  $p$  to state  $q$  if there is a finite sequence of transitions  $\langle (p_0, a_1, p_1), (p_1, a_2, p_2), \dots, (p_{n-1}, a_n, p_n) \rangle \in (Q \times \Sigma \times Q)^*$  with  $p = p_0$  and  $q = p_n$ . We write  $L_{p,q}$  for the set of finite strings  $w$  such that there is a run of  $\mathcal{A}$  over  $w$  from  $p$  to  $q$ . We say that there is a run of  $\mathcal{A}$  over an  $\omega$ -string  $s = a_1 a_2 \dots \in \Sigma^\omega$  if there is a sequence of transitions  $\langle (q_0, a_1, q_1), (q_1, a_2, q_2), \dots \rangle \in (Q \times \Sigma \times Q)^\omega$ . For an infinite run  $r$ , we denote by  $\Omega(r)$  the set of states that occur infinitely often in  $r$ . We say that an  $\omega$ -string  $w$  is accepted by a Muller automaton  $\mathcal{A}$  if the run of  $\mathcal{A}$  on  $w$  is such that  $\Omega(r) \in F$  and we write  $L(\mathcal{A})$  for the set of all  $\omega$ -strings accepted by  $\mathcal{A}$ .

A Muller automaton  $\mathcal{A}$  is *aperiodic* iff there exists some  $m \geq 1$  s.t.  $u^m \in L_{p,q}$  iff  $u^{m+1} \in L_{p,q}$  for all  $u \in \Sigma^*$  and  $p, q \in Q$ . Another equivalent way to define aperiodicity is using the transition monoid, which, to the best of our knowledge, has not been defined in the literature for Muller automata. Given a DMA  $\mathcal{A} = (Q, q_0, \Sigma, \Delta, \{F_1, \dots, F_n\})$ , we define the transition

monoid  $\mathcal{M}_{\mathcal{A}}=(M_{\mathcal{A}}, \times, \mathbf{1})$  of  $\mathcal{A}$  as follows:  $M_{\mathcal{A}}$  is a set of  $|Q| \times |Q|$  square matrices over  $(\{0, 1\} \cup 2^Q)^n \cup \{\perp\}$ . Matrix multiplication  $\times$  is defined for matrices in  $M_{\mathcal{A}}$  with identity element  $\mathbf{1} \in M_{\mathcal{A}}$ , where  $\mathbf{1}$  is the matrix whose diagonal entries are  $(\emptyset, \emptyset, \dots, \emptyset)$  and non-diagonal entries are all  $\perp$ 's. Formally,  $M_{\mathcal{A}} = \{M_s : s \in \Sigma^*\}$  is defined using matrices  $M_s$  for strings  $s \in \Sigma^*$  s.t.  $M_s[p][q] = \perp$  if there is no run from  $p$  to  $q$  over  $s$  in  $\mathcal{A}$ . Otherwise, let  $P$  be the set of states (excluding  $p$  and  $q$ ) witnessed in the unique run from  $p$  to  $q$ . Then  $M_s[p][q] = (x_1, \dots, x_n) \in (\{0, 1\} \cup 2^Q)^n$  where (1)  $x_i = 0$  iff  $\exists t \in P \cup \{p, q\}, t \notin F_i$ ; (2)  $x_i = 1$  iff  $P \cup \{p, q\} = F_i$ , and (3)  $x_i = P \cup \{p, q\}$  iff  $P \cup \{p, q\} \subset F_i$ . It is easy to see that  $M_{\epsilon} = \mathbf{1}$ , since  $\epsilon$  takes a state to itself and nowhere else. The operator  $\times$  is simply matrix multiplication for matrices in  $M_{\mathcal{A}}$ , however we need to define addition  $\oplus$  and multiplication  $\odot$  for elements  $(\{0, 1\} \cup 2^Q)^n \cup \{\perp\}$  of the matrices. We have  $\alpha_1 \odot \alpha_2 = \perp$  if  $\alpha_1 = \perp$  or  $\alpha_2 = \perp$ , and if  $\alpha_1 = (x_1, \dots, x_n)$  and  $\alpha_2 = (y_1, \dots, y_n)$  then  $\alpha_1 \odot \alpha_2 = (z_1, \dots, z_n)$  s.t.:

$$z_i = \begin{cases} 0 & \text{if } x_i = 0 \text{ or } y_i = 0 \\ 1 & \text{if } (x_i = y_i = 1) \text{ or if } (x_i, y_i \subset F_i \text{ and } x_i \cup y_i = F_i) \\ 1 & \text{if } (x_i = 1 \text{ and } y_i \subset F_i) \text{ or } (y_i = 1 \text{ and } x_i \subset F_i) \\ x_i \cup y_i & \text{if } x_i, y_i \subset F_i \text{ and } x_i \cup y_i \subset F_i \end{cases} \quad (\star)$$

Due to determinism, we have that for every matrix  $M_s$  and every state  $p$  there is at most one state  $q$  such that  $M_s[p][q] \neq \perp$  and hence the only addition rule we need to introduce is  $\alpha \oplus \perp = \perp \oplus \alpha = \alpha$ . It is easy to see that  $(M_{\mathcal{A}}, \times, \mathbf{1})$  is a monoid (a proof is deferred to the [11]). It is straightforward to see that a Muller automaton is aperiodic if and only if its transition monoid is aperiodic. [11] gives a proof showing that a language  $L \subseteq \Sigma^\omega$  is aperiodic iff there is an aperiodic DMA accepting it.

### 3 Aperiodic Transformations

In this section we formally introduce three models to express FO-transformations, and prepare the machinery required to prove their expressive equivalence in the rest of the paper.

#### 3.1 First-Order Logic Definable Transformations

Courcelle [10] initiated the study of structure transformations using MSO logic. His main idea was to define a transformation  $(w, w') \in R$  by defining the string model of  $w'$  using a finite number of copies of positions of the string model of  $w$ . The existence of positions, various edges, and position labels are then given as  $\text{MSO}(\Sigma)$  formulas. We study a restriction of his formalism to use first-order logic to express string transformations.

► **Definition 4.** An *FO string transducer* is a tuple  $T=(\Sigma, \Gamma, \phi_{\text{dom}}, C, \phi_{\text{pos}}, \phi_{\preceq})$  where:

- $\Sigma$  and  $\Gamma$  are finite input and output alphabets;
- $\phi_{\text{dom}}$  is a closed  $\text{FO}(\Sigma)$  formula characterizing the domain of the transformation;
- $C = \{1, 2, \dots, n\}$  is a finite index set;
- $\phi_{\text{pos}} = \{\phi_\gamma^c(x) : c \in C \text{ and } \gamma \in \Gamma\}$  is a set of  $\text{FO}(\Sigma)$  formulae with a free variable  $x$ ;
- $\phi_{\preceq} = \{\phi_{\preceq}^{c,d}(x, y) : c, d \in C\}$  is a set of  $\text{FO}(\Sigma)$  formulae with two free variables  $x$  and  $y$ .

The transformation  $\llbracket T \rrbracket$  defined by  $T$  is as follows. A string  $s$  with  $\Xi_s = (\text{dom}(s), \preceq, (L_a)_{a \in \Sigma})$  is in the domain of  $\llbracket T \rrbracket$  if  $s \models \phi_{\text{dom}}$  and the output string  $w$  with structure

$M = (D, \preceq^M, (L_\gamma^M)_{\gamma \in \Gamma})$  is such that

- $D = \{v^c : v \in \text{dom}(s), c \in C \text{ and } \phi^c(v)\}$  is the set of positions where  $\phi^c(v) \stackrel{\text{def}}{=} \bigvee_{\gamma \in \Gamma} \phi_\gamma^c(v)$ ;



- $\preceq^M \subseteq D \times D$  is the ordering relation between positions and it is such that for  $v, u \in \text{dom}(s)$  and  $c, d \in C$  we have that  $v^c \preceq^M u^d$  if  $w \models \phi_{\preceq}^{c,d}(v, u)$ ; and
- for all  $v^c \in D$  we have that  $L_{\gamma}^M(v^c)$  iff  $\phi_{\gamma}^c(v)$ .

Observe that the output is unique and therefore FO transducers implement functions. A string  $s \in \Sigma^\omega$  can be represented by its string-graph with  $\text{dom}(s) = \{i \in \mathbb{N}\}$ ,  $\preceq = \{(i, j) \mid i \leq j\}$  and  $L_a(i)$  iff  $s[i] = a$  for all  $i$ . From now on, we denote the string-graph of  $s$  as  $s$  only. We say that an FO transducer is a *string-to-string* transducer if its domain is restricted to string graphs and the output is also a string graph. We say that a string-to-string transformation is FO-definable if there exists an FO transducer implementing the transformation. We write FOT for the set of FO-definable string-to-string  $\omega$ -transformations.

► **Example 5.** Figure 1(c) shows a transformation for an FOT that implements the transformation  $f_1 : \Sigma^* \# \{a, b\}^\omega \rightarrow \Sigma^\omega$ , where  $\Sigma = \{a, b, \#\}$ , by replacing every maximal  $\#$  free string  $u$  with  $\bar{u}u$ . Let `is_string#` be an FO formula that defines a string that contains a  $\#$ , and let `reach#`( $x$ ) be an FO formula that is true at a position which has a  $\#$  at a later position. To define the FOT formally, we have  $\phi_{\text{dom}} = \text{is\_string}_{\#}$ ,  $\phi_{\gamma}^1(x) = \phi_{\gamma}^2(x) = L_{\gamma}(x) \wedge \neg L_{\#}(x) \wedge \text{reach}_{\#}(x)$ , since we only keep the non  $\#$  symbols that can “reach” a  $\#$  in the input string in the first two copies.  $\phi_{\gamma}^3(x) = L_{\#}(x) \vee (\neg L_{\#}(x) \wedge \neg \text{reach}_{\#}(x))$ , since we only keep the  $\#$ 's, and the infinite suffix from where there are no  $\#$ 's. The full list of formulae  $\phi^{i,j}$  can be seen in [11].

### 3.2 Two-way Transducers (2WST)

A 2WST is a tuple  $T = (Q, \Sigma, \Gamma, q_0, \delta, F)$  where  $\Sigma, \Gamma$  are respectively the input and output alphabet,  $q_0$  is the initial state,  $\delta$  is the transition function and  $F \subseteq 2^Q$  is the acceptance set. The transition function is given by  $\delta : Q \times \Sigma \rightarrow Q \times \Gamma^* \times \{1, 0, -1\}$ . A configuration of the 2WST is a pair  $(q, i)$  where  $q \in Q$  and  $i \in \mathbb{N}$  is the current position of the input string. A run  $r$  of a 2WST on a string  $s \in \Sigma^\omega$  is a sequence of transitions  $(q_0, i_0=0) \xrightarrow{a_1/c_1, \text{dir}}$   $(q_1, i_1) \xrightarrow{a_2/c_2, \text{dir}}$   $(q_2, i_2) \cdots$  where  $a_i \in \Sigma$  is the input letter read and  $c_i \in \Gamma^*$  is the output string produced during a transition and  $i_j$ s are the positions updated during a transition for all  $j \in \text{dom}(s)$ . *dir* is the direction,  $\{1, 0, -1\}$ . W.l.o.g. we can consider the outputs to be over  $\Gamma \cup \{\epsilon\}$ . The output  $\text{out}(r)$  of a run  $r$  is simply a concatenation of the individual outputs, i.e.  $c_1 c_2 \cdots \in \Gamma^\infty$ . We say that the transducer reads the whole string  $s$  when  $\sup \{i_n \mid 0 \leq n < |r|\} = \infty$ . The output of  $s$ , denoted  $T(s)$  is defined as  $\text{out}(r)$  only if  $\Omega(r) \in F$  and  $r$  reads the whole string  $s$ . We write  $\llbracket T \rrbracket$  for the transformation captured by  $T$ .

**Transition Monoid.** The transition monoid of a 2WST  $T = (Q, \Sigma, \Gamma, q_0, \delta, \{F_1, \dots, F_n\})$  is the transition monoid of its underlying automaton. However, since the 2WST can read their input in both directions, the transition monoid definition must allow for reading the string starting from left side and leaving at the left (left-left) and similar other behaviors (left-right, right-left and right-right). Following [9], we define the behaviors  $\mathcal{B}_{xy}(w)$  of a string  $w$  for  $x, y \in \{\ell, r\}$ .  $\mathcal{B}_{\ell r}(w)$  is a set consisting of pairs  $(p, q)$  of states such that starting in state  $p$  in the left side of  $w$  the transducer leaves  $w$  in right side in state  $q$ . In the example in figure 1(a), we have  $\mathcal{B}_{\ell r}(ab\#) = \{(t, t), (p, t), (q, t)\}$  and  $\mathcal{B}_{r r}(ab\#) = \{(q, t), (t, t), (p, q)\}$ . Two words  $w_1, w_2$  are “equivalent” if their left-left, left-right, right-left and right-right behaviors are same. That is,  $\mathcal{B}_{xy}(w_1) = \mathcal{B}_{xy}(w_2)$  for  $x, y \in \{\ell, r\}$ . The transition monoid of  $T$  is the conjunction of the 4 behaviors, which also keeps track, in addition, the set of states witnessed in the run, as shown for the deterministic Muller automata earlier. For each

string  $w \in \Sigma^*$ ,  $x, y \in \{\ell, r\}$ , and states  $p, q$ , the entries of the matrix  $M_u^{xy}[p][q]$  are of the form  $\perp$ , if there is no run from  $p$  to  $q$  on word  $u$ , starting from the side  $x$  of  $u$  and leaving it in side  $y$ , and is  $(x_1, \dots, x_n)$  otherwise, where  $x_i$  is defined exactly as in section 2.3. For equivalent words  $u_1, u_2$ , we have  $M_{u_1}^{xy}[p][q] = M_{u_2}^{xy}[p][q]$  for all  $x, y \in \{\ell, r\}$  and states  $p, q$ . Addition and multiplication of matrices are defined as in the case of Muller automata. See [11] for more details. Note that behavioral composition is quite complex, due to left-right movements. In particular, it can be seen from the example that  $\mathcal{B}_{\ell r}(ab\#a\#) = \mathcal{B}_{\ell r}(ab\#)\mathcal{B}_{\ell\ell}(a\#)\mathcal{B}_{rr}(ab\#)\mathcal{B}_{\ell r}(a\#)$ . Since we assume that the 2WST  $T$  is deterministic and completely reads the input string  $\alpha \in \Sigma^\omega$ , we can find a unique factorization  $\alpha = [\alpha_0 \dots \alpha_{p_1}][\alpha_{p_1+1} \dots \alpha_{p_2}] \dots$  such that the run of  $\mathcal{A}$  on each  $\alpha$ -block progresses from left to right, and each  $\alpha$ -block will be processed completely. That is, one can find a unique sequence of states  $q_{p_1}, q_{p_2}, \dots$  such that the 2WST starting in initial state  $q_0$  at the left of the block  $\alpha_0 \dots \alpha_{p_1}$  leaves it at the right in state  $q_{p_1}$ , starts the next block  $\alpha_{p_1+1} \dots \alpha_{p_2}$  from the left in state  $q_{p_1}$  and leaves it at the right in state  $q_{p_2}$  and so on.

We consider the languages  $L_{pq}^{xy}$  for  $x, y \in \{\ell, r\}$ , where  $\ell, r$  respectively stand for left and right.  $L_{pq}^{\ell\ell}$  stands for all strings  $w$  such that, starting at state  $p$  at the left of  $w$ , one leaves the left of  $w$  in state  $q$ . Similarly,  $L_{pq}^{r\ell}$  stands for all strings  $w$  such that starting at the right of  $w$  in state  $p$ , one leaves the left of  $w$  in state  $q$ . In figure 1(a), note that starting on the right of  $ab\#$  in state  $t$ , we leave it on the right in state  $t$ , while we leave it on the left in state  $p$ . So  $ab\# \in L_{tt}^{rr}, L_{tp}^{r\ell}$ . Also,  $ab\# \in L_{pq}^{rr}$ .

A 2WST is said to be *aperiodic* iff for all strings  $u \in \Sigma^*$ , all states  $p, q$  and  $x, y \in \{\ell, r\}$ , there exists some  $m \geq 1$  such that  $u^m \in L_{pq}^{xy}$  iff  $u^{m+1} \in L_{pq}^{xy}$ .

**Star-Free Lookaround.** We wish to introduce aperiodic 2WST that are capable of capturing FO-definable transformations. However, as we discussed earlier (see page 3 in the paragraph on two-way transducers) 2WST without look-ahead are strictly less expressive than MSO transducers. To remedy this we study aperiodic 2WSTs enriched with star-free look-ahead (star-free look-back can be assumed for free).

An aperiodic 2WST with star-free look-around ( $2WST_{sf}$ ) is a tuple  $(T, A, B)$  where  $A$  is an aperiodic Muller look-ahead automaton and  $B$  is an aperiodic look-behind automaton, resp., and  $T = (\Sigma, \Gamma, Q, q_0, \delta, F)$  is an aperiodic 2WST as defined earlier except that the transition function  $\delta : Q \times Q_B \times \Sigma \times Q_A \rightarrow Q \times \Gamma \times \{-1, 0, +1\}$  may consult look-ahead and look-behind automata to make its decisions. Let  $s \in \Sigma^\omega$  be an input string, and  $L(A, p)$  be the set of infinite strings accepted by  $A$  starting in state  $p$ . Similarly, let  $L(B, r)$  be the set of finite strings accepted by  $B$  starting in state  $r$ . We assume that  $2WST_{sf}$  are *deterministic* i.e. for every string  $s \in \Sigma^\omega$  and every input position  $i \leq |s|$ , there is exactly one state  $p \in Q_A$  and one state  $r \in Q_B$  such that  $s(i)s(i+1) \dots \in L(A, p)$  and  $s(0)s(1) \dots s(i-1) \in L(B, r)$ . If the current configuration is  $(q, i)$  and  $\delta(q, r, s(i), p) = (q', z, d)$  is a transition, such that the string  $s(i)s(i+1) \dots \in L(A, p)$  and  $s(0)s(1) \dots s(i-1) \in L(B, r)$ , then  $2WST_{sf}$  writes  $z \in \Gamma$  on the output tape and updates its configuration to  $(q', i+d)$ . Figure 1(a) shows a 2WST with star-free look-ahead  $\text{reach}_\#(x)$  capturing the transformation  $f_1$  (details in [11]).

### 3.3 Streaming $\omega$ -String Transducers (SST)

Streaming string transducers (SSTs) manipulate a finite set of string variables to compute their output. In this section we introduce aperiodic SSTs for infinite strings. Let  $\mathcal{X}$  be a finite set of variables and  $\Gamma$  be a finite alphabet. A substitution  $\sigma$  is defined as a mapping  $\sigma : \mathcal{X} \rightarrow (\Gamma \cup \mathcal{X})^*$ . A valuation is defined as a substitution  $\sigma : \mathcal{X} \rightarrow \Gamma^*$ . Let  $\mathcal{S}_{\mathcal{X}, \Gamma}$  be the set of all substitutions  $[\mathcal{X} \rightarrow (\Gamma \cup \mathcal{X})^*]$ . Any substitution  $\sigma$  can be extended to  $\hat{\sigma} : (\Gamma \cup \mathcal{X})^* \rightarrow$



$(\Gamma \cup \mathcal{X})^*$  in a straightforward manner. The composition  $\sigma_1\sigma_2$  of two substitutions  $\sigma_1$  and  $\sigma_2$  is defined as the standard function composition  $\hat{\sigma}_1\sigma_2$ , i.e.  $\hat{\sigma}_1\sigma_2(x) = \hat{\sigma}_1(\sigma_2(x))$  for all  $x \in \mathcal{X}$ . We say that a string  $u \in (\Gamma \cup \mathcal{X})^*$  is *copyless* (or linear) if each  $x \in \mathcal{X}$  occurs at most once in  $u$ . A substitution  $\sigma$  is copyless if  $\hat{\sigma}(u)$  is copyless, for all linear  $u \in (\Gamma \cup \mathcal{X})^*$ .

► **Definition 6.** A *streaming  $\omega$ -string transducer* (SST) is a tuple  $T = (\Sigma, \Gamma, Q, q_0, \delta, \mathcal{X}, \rho, F)$

- $\Sigma$  and  $\Gamma$  are finite input and output alphabets;
- $Q$  is a finite set of states with initial state  $q_0$ ;
- $\delta : Q \times \Sigma \rightarrow Q$  is a transition function and  $\mathcal{X}$  is a finite set of variables;
- $\rho : (Q \times \Sigma) \rightarrow \mathcal{S}_{\mathcal{X}, \Gamma}$  is a variable update function to copyless substitutions such that any variable  $x$  occurs at most once on the right hand side of a simultaneous substitution;
- $F : 2^Q \rightarrow \mathcal{X}^*$  is an output function such that for all  $P \in \text{dom}(F)$  the string  $F(P)$  is copyless of form  $x_1 \dots x_n$ , and for  $q, q' \in P$  and  $a \in \Sigma$  s.t.  $q' = \delta(q, a)$  we have
  - $\rho(q, a)(x_i) = x_i$  for all  $i < n$  and  $\rho(q, a)(x_n) = x_n u$  for some  $u \in (\Gamma \cup \mathcal{X})^*$ .

The concept of a run of an SST is defined in an analogous manner to that of a Muller automaton. The sequence  $\langle \sigma_{r,i} \rangle_{0 \leq i \leq |r|}$  of substitutions induced by a run  $r = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \dots$  is defined inductively as the following:  $\sigma_{r,i} = \sigma_{r,i-1} \rho(q_{i-1}, a_i)$  for  $0 < i \leq |r|$  and  $\sigma_{r,0} = x \in X \mapsto \varepsilon$ . The output  $T(r)$  of an infinite run  $r$  of  $T$  is defined only if  $F(r)$  is defined and equals  $T(r) \stackrel{\text{def}}{=} \lim_{i \rightarrow \infty} \langle \sigma_{r,i}(F(r)) \rangle$ , when the limit exists. If not, we pad  $\perp^\omega$  to the obtained finite string to get  $\lim_{i \rightarrow \infty} \langle \sigma_{r,i}(F(r)) \perp^\omega \rangle$  as the infinite output string.

The assumptions on the output function  $F$  in the definition of an SST ensure that this limit exists whenever  $F(r)$  is defined. Indeed, when a run  $r$  reaches a point from where it visits only states in  $P$ , these assumptions enforce the successive valuations of  $F(P)$  to be an increasing sequence of strings by the prefix relation. The padding by unique letter  $\perp$  ensures that the output is always an  $\omega$ -string. The output  $T(s)$  of a string  $s$  is then defined as the output  $T(r)$  of its unique run  $r$ . The transformation  $\llbracket T \rrbracket$  defined by an SST  $T$  is the partial function  $\{(s, T(s)) : T(s) \text{ is defined}\}$ . See [11] for an example. We remark that for every SST  $T = (\Sigma, \Gamma, Q, q_0, \delta, \mathcal{X}, \rho, F)$ , its domain is always an  $\omega$ -regular language defined by the Muller automaton  $(\Sigma, Q, q_0, \delta, \text{dom}(F))$ , which can be constructed in linear time. However, the range of an SST may not be  $\omega$ -regular. For instance, the range of the SST-definable transformation  $a^n \#^\omega \mapsto a^n b^n \#^\omega$  ( $n \geq 0$ ) is not  $\omega$ -regular.

**Aperiodic Streaming String Transducers.** We define the notion of aperiodic SSTs by introducing an appropriate notion of transition monoid for transducers. The transition monoid of an SST  $T$  is based on the effect of a string  $s$  on the states as well as on the variables. The effect on variables is characterized by, what we call, flow information that is given as a relation that describes the number of copies of the content of a given variable that contribute to another variable after reading a string  $s$ .

Let  $T = (\Sigma, \Gamma, Q, q_0, \delta, \mathcal{X}, \rho, F)$  be an SST. Let  $s$  be a string in  $\Sigma^*$  and suppose that there exists a run  $r$  of  $T$  on  $s$ . Recall that this run induces a substitution  $\sigma_r$  that maps each variable  $X \in \mathcal{X}$  to a string  $u \in (\Gamma \cup \mathcal{X})^*$ . For string variables  $X, Y \in \mathcal{X}$ , states  $p, q \in Q$ , and  $n \in \mathbb{N}$  we say that  $n$  copies of  $Y$  flow to  $X$  from  $p$  to  $q$  if there exists a run  $r$  on  $s \in \Sigma^*$  from  $p$  to  $q$ , and  $Y$  occurs  $n$  times in  $\sigma_r(X)$ . We extend the notion of transition monoid for the Muller automata as defined in Section 2 for the transition monoid for SSTs to equip it with variables. Formally, the transition monoid  $\mathcal{M}_T = (M_T, \times, \mathbf{1})$  of an SST  $T = (\Sigma, \Gamma, Q, q_0, \delta, \mathcal{X}, \rho, \{F_1, \dots, F_n\})$  is such that  $M_T$  is a set of  $|Q \times \mathcal{X}| \times |Q \times \mathcal{X}|$  square matrices over  $(\mathbb{N} \times (\{0, 1\} \cup 2^Q)^n) \cup \{\perp\}$  along with matrix multiplication  $\times$  defined for matrices in  $M_T$  and identity element  $\mathbf{1} \in M_T$  is the matrix whose diagonal entries are  $(1, (\emptyset, \emptyset, \dots, \emptyset))$  and non-diagonal entries are all  $\perp$ 's. Formally  $M_T = \{M_s : s \in \Sigma^*\}$  is defined using matrices

$M_s$  for strings  $s \in \Sigma^*$  s.t.  $M_s[(p, X)][(q, Y)] = \perp$  if there is no run from state  $p$  to state  $q$  over  $s$  in  $T$ , otherwise  $M_s[(p, X)][(q, Y)] = (k, (x_1, \dots, x_n)) \in (\mathbb{N} \times (\{0, 1\} \cup 2^Q)^n)$  where  $x_i$  is defined exactly as in section 2.3, and  $k$  copies of variable  $X$  flow to variable  $Y$  from state  $p$  to state  $q$  after reading  $s$ .

We write  $(p, X) \rightsquigarrow_\alpha^u (q, Y)$  for  $M_u[(p, X)][(q, Y)] = \alpha$ .

It is easy to see that  $M_\epsilon = \mathbf{1}$ . The operator  $\times$  is simply matrix multiplication for matrices in  $M_T$ , however we need to define addition  $\oplus$  and multiplication  $\odot$  for elements  $(\{0, 1\} \cup 2^Q)^n \cup \{\perp\}$  of the matrices. We have  $\alpha_1 \odot \alpha_2 = \perp$  if  $\alpha_1 = \perp$  or  $\alpha_2 = \perp$ , and if  $\alpha_1 = (k_1, (x_1, \dots, x_n))$  and  $\alpha_2 = (k_2, (y_1, \dots, y_n))$  then  $\alpha_1 \odot \alpha_2 = (k_1 \times k_2, (z_1, \dots, z_n))$  s.t. for all  $1 \leq i \leq n$   $z_i$  are defined as in  $(\star)$  from Section 2.3. Note that due to determinism of the SSTs we have that for every matrix  $M_s$  and every state  $p$  there is at most one state  $q$  such that  $M_s[p][q] \neq \perp$  and hence the only addition rules we need to introduce is  $\alpha \oplus \perp = \perp \oplus \alpha = \alpha$ ,  $0 \oplus 0 = 0$ ,  $1 \oplus 1 = 1$  and  $\kappa \oplus \kappa = \kappa$  for  $\kappa \subseteq Q$ . It is easy to see that  $(M_T, \times, \mathbf{1})$  is a monoid and we give a proof in [11]. We say that the transition monoid  $M_T$  of an SST  $T$  is 1-bounded if in all entries  $(j, (x_1, \dots, x_n))$  of the matrices of  $M_T$ ,  $j \leq 1$ . A streaming string transducer is *aperiodic* if its transition monoid is aperiodic.

#### 4 FOTs $\equiv$ Aperiodic 2WST<sub>sf</sub>

► **Theorem 7.** *A transformation  $f : \Sigma^\omega \rightarrow \Gamma^\omega$  is FOT-definable if and only if it is definable using an aperiodic two way transducer with star-free look-around.*

**Proof (Sketch).** This proof is in two parts.

- **Aperiodic 2WST<sub>sf</sub>  $\subseteq$  FOT.** We first show that given an aperiodic 2WST<sub>sf</sub>  $\mathcal{A}$ , we can effectively construct an FOT that captures the same transduction as  $\mathcal{A}$  over infinite words. Let  $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \delta, F)$  be an aperiodic 2WST<sub>sf</sub>, where each transition outputs at most one letter. Note that this is without loss of generality, since we can output any longer string by having some extra states. Given  $\mathcal{A}$ , we construct the FOT  $T = (\Sigma, \Gamma, \phi_{dom}, C, \phi_{pos}, \phi_{\prec})$  that realizes the transduction of  $\mathcal{A}$ . The formula  $\phi_{dom}$  is the conjunction of formulae `is_string` and  $\varphi$  where  $\varphi$  is a FO formula that captures the set of accepted strings of  $\mathcal{A}$  (obtained by proving  $L(\mathcal{A})$  is aperiodic [11]) and `is_string` is a FO formula that specifies that the input graph is a string (see [11]). The copies of the FOT are the states of  $\mathcal{A}$ . For any two positions  $x, y$  of the input string, and any two copies  $q, q'$ , we need to define  $\phi_{\prec}^{q, q'}$ . This is simply describing the behaviour of  $\mathcal{A}$  on the substring from position  $x$  to position  $y$  of the input string  $u$ , assuming at position  $x$ , we are in state  $q$ , and reach state  $q'$  at position  $y$ . The following lemma (proof in [11]) gives an FO formula  $\psi_{q, q'}(x, y)$  describing this.

► **Lemma 8.** *Let  $\mathcal{A}$  be an aperiodic 2WST<sub>sf</sub> with the Muller acceptance condition. Then for all pairs of states  $q, q'$ , there exists an FO formula  $\psi_{q, q'}(x, y)$  such that for all strings  $s \in \Sigma^\omega$  and a pair of positions  $x, y$  of  $s$ ,  $s \models \psi_{q, q'}(x, y)$  iff there is a run from state  $q$  starting at position  $x$  of  $s$  that reaches position  $y$  of  $s$  in state  $q'$ .*

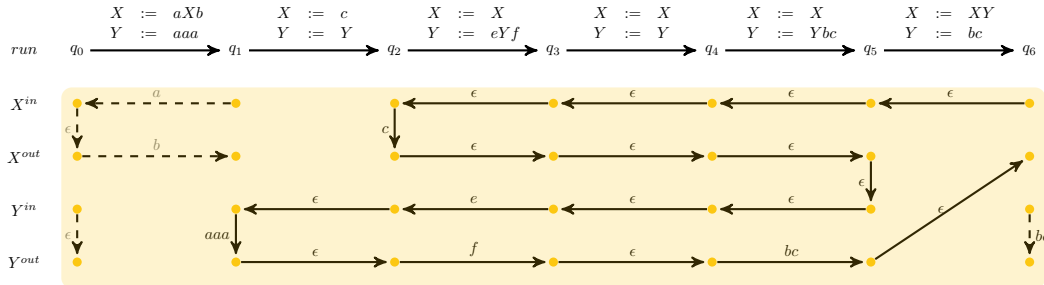
An edge exists between position  $x$  of copy  $q$  and position  $y$  of copy  $q'$  iff the input string  $u \models \psi_{q, q'}(x, y)$ . The formulae  $\phi_\gamma^q(x)$  for each copy  $q$  specifies the output at position  $x$  in state  $q$ . We have to capture that position  $x$  is reached from the initial position in state  $q$ , and also the possible outputs produced while in state  $q$  at  $x$ . The transition function  $\delta$  gives us these symbols. The formula  $\bigvee_{\delta(q, a) = (q', dir, \gamma)} L_a(x)$  captures the possible output symbols. To state that we reach  $q$  at position  $x$ , we say  $\exists y[\text{first}(y) \wedge \psi_{q_0, q}(y, x)]$ . The conjunction of these two formulae gives  $\phi_\gamma^q(x)$ . This completes the FOT  $T$ .

- **FOT  $\subseteq$  Aperiodic  $2WST_{sf}$ .** Given an FOT, we show that we can construct an aperiodic  $2WST$  with star-free look-around capturing the same transduction over  $\omega$ -words. For this, we first show that given an FOT, we can construct  $2WST$  enriched with FO instructions that captures the same transduction as the FOT. The idea of the proof follows [13], where one first defines an intermediate model of aperiodic  $2WST$  with FO instructions instead of look-around. Then we show  $FOT \subseteq 2WST_{fo} \subseteq 2WST_{sf}$ , to complete the proof. The omitted details can be found in [11]. ◀

## 5 Aperiodic SST $\subset$ FOT

► **Lemma 9.** *A transformation is FO-definable if it is aperiodic-SST definable.*

We show that every aperiodic 1-bounded SST definable transformation is definable using FO-transducers. A crucial component in the proof of this lemma is to show that the variable flow in the aperiodic 1-bounded SST is FO-definable ([11]). To construct the FOT, we make use of the output structure for SST. It is an intermediate representation of the output, and the transformation of any input string into its SST-output structure will be shown to be FO-definable. For any SST  $T$  and string  $s \in \text{dom}(T)$ , the SST-output structure of  $s$  is a relational structure  $G_T(s)$  obtained by taking, for each variable  $X \in \mathcal{X}$ , two copies of  $\text{dom}(s)$ , respectively denoted by  $X^{in}$  and  $X^{out}$ . For notational convenience we assume that these structures are labeled on the edges. A pair  $(X, i)$  is *useful* if the content of variable  $X$  before reading  $s[i]$  will be part of the output after reading the whole string  $s$ . This structure satisfies the following *invariants*: for all  $i \in \text{dom}(s)$ , (1) the nodes  $(X^{in}, i)$  and  $(X^{out}, i)$  exist only if  $(X, i)$  is useful, and (2) there is a directed path from  $(X^{in}, i)$  to  $(X^{out}, i)$  whose labels are same as variable  $X$  computed by  $T$  after reading  $s[i]$ .



We define SST-output structures formally in [11], however, the illustration above shows an SST-output structure. We show only the variable updates. Dashed arrows represent variable updates for useless variables, and therefore does not belong the SST-output structure. The path from  $(X^{in}, 6)$  to  $(X^{out}, 6)$  gives the contents of  $X$  ( $ceaaaabc$ ) after 6 steps. We write  $O_T$  for the set of strings appearing in right-hand side of variable updates.

We next show that the transformation that maps an  $\omega$ -string  $s$  into its output structure is FO-definable, whenever the SST is 1-bounded and aperiodic. Using the fact that variable flow is FO-definable, we show that for any two variables  $X, Y$ , we can capture in FO, a path from  $(X^d, i)$  to  $(Y^e, j)$  for  $d, e \in \{in, out\}$  in  $G_T(s)$  and all positions  $i, j$ .

► **Lemma 10.** *Let  $T$  be an **aperiodic, 1-bounded** SST  $T$ . For all  $X, Y \in \mathcal{X}$  and all  $d, d' \in \{in, out\}$ , there exists an  $FO[\Sigma]$ -formula  $path_{X,Y,d,d'}(x, y)$  with two free variables such that for all strings  $s \in \text{dom}(T)$  and all positions  $i, j \in \text{dom}(s)$ ,  $s \models path_{X,Y,d,d'}(i, j)$  iff there exists a path from  $(X^d, i)$  to  $(Y^{d'}, j)$  in  $G_T(s)$ .*

The proof of Lemma 10 is in longer version [11]. As seen in [11] (in Proposition 4) one can write a formula  $\phi_q(x)$  (to capture the state  $q$  reached) and formula  $\psi_P^{Rec}$  (to capture the recurrence of a Muller set  $P$ ) in an accepting run after reading a prefix. For each variable  $X \in \mathcal{X}$ , we have two copies  $X^{in}$  and  $X^{out}$  that serve as the copy set of the FOT. As given by the SST output-structure, for each step  $i$ , state  $q$  and symbol  $a$ , a copy is connected to copies in the previous step based on the updates  $\rho(q, a)$ . The full details of the FOT construction handling the Muller acceptance condition of the SST are in [11].

## 6 Aperiodic $2WST_{sf} \subset$ Aperiodic SST

We show that given an aperiodic  $2WST \mathcal{A} = (\Sigma, \Gamma, Q, q_0, \delta, F)$  with star-free look around over  $\omega$ -words, we can construct an aperiodic SST  $\mathcal{T}$  that realizes the same transformation.

► **Lemma 11.** *For every transformation definable with an aperiodic  $2WST$  with star-free look around, there exists an equivalent aperiodic 1-bounded SST.*

**Proof.** While the idea of the construction is similar to [3], the main challenge is to eliminate the star-free look-around for infinite strings from the SST, preserving aperiodicity. As an intermediate model we introduce streaming  $\omega$ -string transducers with star-free look-around  $SST_{sf}$  that can make transitions based on some star-free property of the input string. We first show that for every aperiodic  $2WST_{sf}$  one can obtain an aperiodic  $SST_{sf}$ , and then prove that the star-free look arounds can be eliminated from the  $SST_{sf}$ .

- ( $2WST_{sf} \subset SST_{sf}$ ). One of the key observations in the construction is that a  $2WST_{sf}$  can move in either direction, while  $SST_{sf}$  cannot. Since we start with a deterministic  $2WST_{sf}$  that reads the entire input string, it is clear that if a cell  $i$  is visited in a state  $q$ , then we never come back to that cell in the same state. We keep track in each cell  $i$ , with current state  $q$ , the state  $f(q)$  the  $2WST_{sf}$  will be in, when it moves into cell  $i + 1$  for the first time. The  $SST_{sf}$  will move from state  $q$  in cell  $i$  to state  $f(q)$  in cell  $i + 1$ , keeping track of the output produced in the interim time; that is, the output produced between  $q$  in cell  $i$  and  $f(q)$  in cell  $i + 1$  must be produced by the  $SST_{sf}$  during the move. This output is stored in a variable  $X_q$ . The state of the  $SST_{sf}$  at each point of time thus comprises of a pair  $(q, f)$  where  $q$  is the current state of the  $2WST_{sf}$ , and  $f$  is the function which computes the state that  $q$  will evolve into, when moving to the right, the first time. In each cell  $i$ , the state of the SST will coincide with the state the  $2WST_{sf}$  is in, when reading cell  $i$  for the first time. In particular, in the  $SST_{sf}$ , we define  $\delta'((q, f), r, a, p) = (f'(q), f')$  where  $f'(q) = f'(f(t))$  if in the  $2WST_{sf}$  we have  $\delta(q, r, a, p) = (t, \gamma, -1)$ .  $f'(q)$  gives the state in which the  $2WST_{sf}$  will move to the right of the current cell, but clearly this depends on  $f(t)$ , the state in which the  $2WST_{sf}$  will move to the right from the previous cell. The variables of the  $SST_{sf}$  are of the form  $X_q$ , where  $q$  is the current state of the  $SST_{sf}$ . Update of  $X_q$  depends on whether the  $2WST_{sf}$  moves left, right or stays in state  $q$ . For example,  $X_q$  is updated as  $X_t \rho(X_{f(t)})$  if in the  $2WST$ ,  $\delta(q, r, a, p) = (t, \gamma, -1)$  and  $f(t)$  is defined. The definition is recursive, and  $X_t$  handles the output produced from state  $t$  in cell  $i - 1$ . We allow all subsets of  $Q$  as Muller sets of the  $SST_{sf}$ , and keep any checks on these, as part of the look-ahead.

A special variable  $O$  is used to define the output of the Muller sets, by simply updating it as  $O := O\rho(X_q)$  corresponding to the current state  $q$  of the  $2WST_{sf}$  (and  $(q, f)$  is the state of the  $SST_{sf}$ ). The details of the correctness of construction are in [11].

- ( $SST_{sf} \subset SST$ ). An aperiodic SST with star-free lookaround is a tuple  $(T, B, A)$  where  $A = (P_A, \Sigma, \delta_A, P_f)$  is an aperiodic, deterministic Muller automaton called a look-ahead

automaton,  $B = (P_B, \Sigma, \delta_B)$  is an aperiodic automaton called the look-behind automaton, and  $T$  is a tuple  $(\Sigma, \Gamma, Q, q_0, \delta, \mathcal{X}, \rho, F)$  where  $\Sigma, \Gamma, Q, q_0, \mathcal{X}, \rho,$  and  $F$  are defined in the same fashion as for  $\omega$ -SSTs, and  $\delta : Q \times P_B \times \Sigma \times P_A \rightarrow Q$  is the transition function. On a string  $a_1 a_2 \dots$ , while processing symbol  $a_i$ , we have in the  $\text{SST}_{sf}$ ,  $\delta((q, p_B, p_A), a_i) = q'$ , (and the next transition is  $\delta((q', p'_B, p'_A), a_{i+1})$ ) if (i) the prefix  $a_1 a_2 \dots a_i \in L(p_A)$ , (ii) the suffix  $a_{i+1} a_{i+2} \dots \in L(p_B)$ , where  $L(p_A)$  ( $L(p_B)$ ) denotes the language accepted starting in state  $p_A$  ( $p_B$ ). We further assume that the look-aheads are mutually exclusive, i.e. for all symbols  $a \in \Sigma$ , all states  $q \in Q$ , and all transitions  $q' = \delta(q, r, a, p)$  and  $q'' = \delta(q, r', a, p')$ , we have that  $L(A_p) \cap L(A_{p'}) = \emptyset$  and  $L(B_r) \cap L(B_{r'}) = \emptyset$ . In [11], we show that for any input string, there is at most one useful, accepting run in the  $\text{SST}_{sf}$ , while in Lemma 29 in [11], we show that adding (aperiodic) look-arounds to SST does not increase their expressiveness.

The proof sketch is now complete.  $\blacktriangleleft$

## 7 Conclusion

We extended the notion of aperiodicity from finite string transformations to that on infinite strings. We have shown a way to generalize transition monoids for deterministic Muller automata to streaming string transducers and two-way finite state transducers that capture the FO definable global transformations. An interesting and natural next step is to investigate LTL-definable transformations, their connection with FO-definable transformations, and their practical applications in verification and synthesis.

---

### References

- 1 R. Alur, A. Durand-Gasselin, and A. Trivedi. From monadic second-order definable string transformations to transducers. In *LICS*, pages 458–467, 2013.
- 2 Rajeev Alur and Loris D’Antoni. *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, chapter Streaming Tree Transducers, pages 42–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- 3 Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. Regular transformations of infinite strings. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science, LICS ’12*, pages 65–74, Washington, DC, USA, 2012. IEEE Computer Society.
- 4 Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, May 2009.
- 5 Rajeev Alur and Pavol Černý. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL ’11*, pages 599–610. ACM, 2011.
- 6 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–12, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 7 J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960.

- 8 J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Int. Congr. for Logic Methodology and Philosophy of Science*, pages 1–11. Stanford University Press, Stanford, 1962.
- 9 Olivier Carton and Luc Dartois. Aperiodic two-way transducers and fo-transductions. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, pages 160–174, 2015.
- 10 B. Courcelle. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, 126(1):53–75, 1994.
- 11 Vrunda Dave, Shankara Narayanan Krishna, and Ashutosh Trivedi. Fo-definable transformations of infinite strings. *CoRR*, abs/1607.04910, 2016.
- 12 V. Diekert and P. Gastin. First-order definable languages. pages 261–306, 2008.
- 13 J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, 2:216–254, 2001.
- 14 Emmanuel Filiot. *Logic and Its Applications: 6th Indian Conference, ICLA 2015, Mumbai, India, January 8-10, 2015. Proceedings*, chapter Logic-Automata Connections for Transformations, pages 30–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
- 15 Emmanuel Filiot, Shankara Narayanan Krishna, and Ashutosh Trivedi. First-order definable string transformations. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 147–159, 2014.
- 16 Richard E. Ladner. Application of model theoretic games to discrete linear orders and finite automata. *Information and Control*, 33(4):281–303, 1977.
- 17 R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Inform. Contr.*, 9:521–530, 1966.
- 18 R. McNaughton and S. Papert. Counter-free automata. *M.I.T. Research Monograph*, 65, 1971. With an appendix by William Henneman.
- 19 Dominique Perrin. *Mathematical Foundations of Computer Science 1984: Proceedings, 11th Symposium Praha, Czechoslovakia September 3–7, 1984*, chapter Recent results on automata and infinite words, pages 134–148. Springer Berlin Heidelberg, 1984.
- 20 M.P. Schuetzenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190 – 194, 1965.
- 21 H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, 1994.
- 22 W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, pages 389–455. Springer, 1996.
- 23 W. Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *In Structures in Logic and Computer Science: A Selection of Essays in Honor of A. Ehrenfeucht, Lecture*, pages 118–143. Springer-Verlag, 1997.
- 24 Wolfgang Thomas. Star-free regular sets of  $\omega$ -sequences. *Information and Control*, 42(2):148–156, 1979.