Aperiodic String Transducers *

Luc Dartois[1], Ismaël Jecker[1], Pierre-Alain Reynier[2]

[1] Université Libre de Bruxelles, Belgium
[2] Aix-Marseille Université, CNRS, LIF UMR 7279, France

Abstract. Regular string-to-string functions enjoy a nice triple characterization through deterministic two-way transducers (2DFT), streaming string transducers (SST) and MSO definable functions. This result has recently been lifted to FO definable functions, with equivalent representations by means of *aperiodic* 2DFT and *aperiodic* 1-bounded SST, extending a well-known result on regular languages. In this paper, we give three direct transformations: *i*) from 1-bounded SST to 2DFT, *ii*) from 2DFT to copyless SST, and *iii*) from k-bounded to 1-bounded SST. We give the complexity of each construction and also prove that they preserve the aperiodicity of transducers. As corollaries, we obtain that FO definable string-to-string functions are equivalent to SST whose transition monoid is finite and aperiodic, and to aperiodic copyless SST.

1 Introduction

The theory of regular languages constitutes a cornerstone in theoretical computer science. Initially studied on languages of finite words, it has since been extended in numerous directions, including finite and infinite trees. Another natural extension is moving from languages to transductions. We are interested in this work in string-to-string transductions, and more precisely in string-to-string functions. One of the strengths of the class of regular languages is their equivalent presentation by means of automata, logic, algebra and regular expressions. The class of so-called *regular string functions* enjoys a similar multiple presentation. It can indeed be alternatively defined using deterministic two-way finite state transducers (2DFT), using Monadic Second-Order graph transductions interpreted on strings (MSOT) [9], and using the model of streaming string transducers (SST) [1]. More precisely, regular string functions are equivalent to different classes of SST, namely copyless SST [1] and k-bounded SST, for every positive integer k [3]. Different papers [9, 1, 3, 2] have proposed transformations between 2DFT, MSOT and SST, summarized on Figure 1.

The connection between automata and logic, which has been very fruitful for model-checking for instance, also needs to be investigated in the framework of transductions. As it has been done for regular languages, an important objective is then to provide similar logic-automata connections for subclasses of regular functions, providing decidability results for these subclasses. As an illustration,

* This work is supported by the ARC project Transform (French speaking community of Belgium), the Belgian FNRS PDR project Flare, and the PHC project VAST (35961QJ) funded by Campus France and WBI.

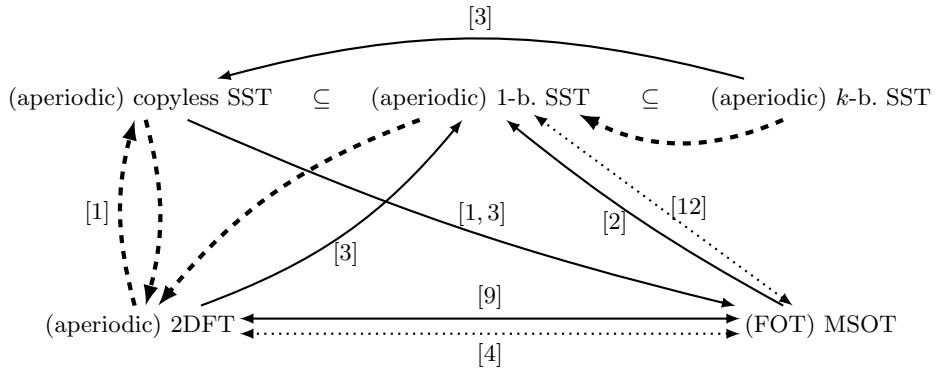


Fig. 1: Summary of transformations between equivalent models. k-b. stands for k-bounded. Plain (resp. dotted) arrows concern regular models (resp. bracketed models). Original constructions presented in this paper are depicted by thick dashed arrows and are valid for both regular and aperiodic versions of the models.

the class of rational functions (accepted by one-way finite state transducers) owns a simple characterization in terms of logic, as shown in [10]. The corresponding logical fragment is called order-preserving MSOT. The decidability of the one-way definability of a two-way transducer proved in [11] thus yields the decidability of this fragment inside the class of MSOT.

The first-order logic considered with order predicate constitutes an important fragment of the monadic second order logic. It is well known that languages definable using this logic are equivalent to those recognized by finite state automata whose transition monoid is aperiodic (as well as other models such as star-free regular expressions). These positive results have motivated the study of similar connections between first-order definable string transformations (FOT) and restrictions of state-based transducers models. Two recent works provide such characterizations for 1-bounded SST and 2DFT respectively [12, 4]. To this end, the authors study a notion of transition monoid for these transducers, and prove that FOT is expressively equivalent to transducers whose transition monoid is aperiodic by providing back and forth transformations between FOT and 1-bounded aperiodic SST (resp. aperiodic 2DFT). In particular, [12] lets as an open problem whether FOT is also equivalent to aperiodic copyless SST and to aperiodic k-bounded SST, for every positive integer k. It is also worth noticing that these characterizations of FOT, unlike the case of languages, do not allow to decide the class FOT inside the class MSOT. Indeed, while decidability for languages relies on the syntactic congruence of the language, no such canonical object exists for the class of regular string transductions.

In this work, we aim at improving our understanding of the relationships between 2DFT and SST. We first provide an original transformation from 1-bounded (or copyless) SST to 2DFT, and study its complexity. While the existing construction used MSO transformations as an intermediate formalism, resulting in a non-elementary complexity, our construction is in double exponential time,

and in single exponential time if the input SST is copyless. Conversely, we describe a direct construction from 2DFT to copyless SST, which is similar to that of [1], but avoids the use of an intermediate model. These constructions also allow to establish links between the crossing degree of a 2DFT, and the number of variables of an equivalent copyless (resp. 1-bounded) SST, and conversely. Last, we provide a direct construction from k-bounded SST to 1-bounded SST, while the existing one was using copyless SST as a target model and not 1-bounded SST [3]. These constructions are represented by thick dashed arrows on Figure 1.

In order to lift these constructions to aperiodic transducers, we introduce a new transition monoid for SST, which is intuitively more precise than the existing one (more formally, the existing one divides the one we introduce). We use this new monoid to prove that the three constructions we have considered above preserve the aperiodicity of the transducer. As a corollary, this implies that FOT is equivalent to both aperiodic copyless and k-bounded SST, for every integer k, two results that were stated as conjectures in [12] (see Figure 1).

Omitted proofs can be found in [7].

2 Definitions

2.1 Words, Languages and Transducers

Given a finite alphabet A, we denote by A^* the set of finite words over A, and by ϵ the empty word. The length of a word $u \in A^*$ is its number of symbols, denoted by $|u|$. For all $i \in \{1, \dots, |u|\}$, we denote by $u[i]$ the i-th letter of u.

A *language* over A is a set $L \subseteq A^*$. Given two alphabets A and B, a *transduction* from A to B is a relation $R \subseteq A^* \times B^*$. A transduction R is *functional* if it is a function. The transducers we will introduce will define transductions. We will say that two transducers T, T' are equivalent whenever they define the same transduction.

Automata A *deterministic two-way finite state automaton* (2DFA) over a finite alphabet A is a tuple $\mathcal{A} = (Q, q_0, F, \delta)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and δ is the transition function, of type $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \to Q \times \{+1, 0, -1\}$. The new symbols $\vdash$ and $\dashv$ are called *endmarkers*.

An input word u is given enriched by the endmarkers, meaning that $\mathcal{A}$ reads the input $\vdash u \dashv$. We set $u[0] = \vdash$ and $u[|u| + 1] = \dashv$. Initially the head of $\mathcal{A}$ is on the first cell $\vdash$ in state q_0 (the cell at position 0). When $\mathcal{A}$ reads an input symbol, depending on the transitions in Δ, its head moves to the left (-1), or stays at the same position (0), or moves to the right ($+1$). To ensure the fact that the reading of $\mathcal{A}$ does not go out of bounds, we assume that there is no transition moving to the left (resp. to the right) on input symbol $\vdash$ (resp. $\dashv$). $\mathcal{A}$ stops as soon as it reaches the endmarker $\dashv$ in a final state.

A *configuration* of $\mathcal{A}$ is a pair $(q, i) \in Q \times \mathbb{N}$ where q is a state and i is a position on the input tape. A *run* ρ of $\mathcal{A}$ is a finite sequence of configurations. The run $\rho = (p_1, i_1) \dots (p_m, i_m)$ is a run on an input word $u \in A^*$ of length n if $i_m \leqslant n+1$,

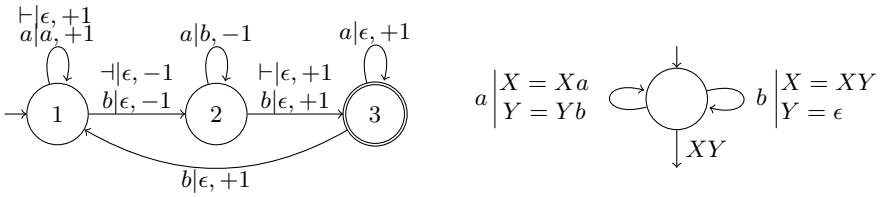


Fig. 2: Aperiodic 2DFT (left) and SST (right) realizing the function f.

and for all $k \in \{1, \dots, m-1\}$, $0 \leqslant i_k \leqslant n+1$ and $(p_k, u[i_k], p_{k+1}, i_{k+1} - i_k) \in \Delta$. It is *accepting* if $p_1 = q_0$, $i_1 = 0$, and m is the only index where both $i_m = n+1$ and $p_m \in F$. The language of a 2DFA $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of words u such that there exists an accepting run of $\mathcal{A}$ on u.

Transducers *Deterministic two-way finite state transducers* (2DFT) over A extend 2DFA with a one-way left-to-right output tape. They are defined as 2DFA except that the transition relation δ is extended with outputs: $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \to B^* \times Q \times \{-1, 0, +1\}$. When a transition (q, a, v, q', m) is fired, the word v is appended to the right of the output tape.

A run of a 2DFT is a run of its underlying automaton, i.e. the 2DFA obtained by ignoring the output (called its *underlying input automaton*). A run ρ may be simultaneously a run on a word u and on a word $u' \neq u$. However, when the input word is given, there is a unique sequence of transitions associated with ρ. Given a 2DFT T, an input word $u \in A^*$ and a run $\rho = (p_1, i_1) \dots (p_m, i_m)$ of T on u, the output of ρ on u is the word obtained by concatenating the outputs of the transitions followed by ρ. If ρ contains a single configuration, this output is simply ϵ. The transduction defined by T is the relation $R(T)$ defined as the set of pairs $(u, v) \in A^* \times B^*$ such that v is the output of an accepting run ρ on the word u. As T is deterministic, such a run is unique, thus $R(T)$ is a function.

Streaming String Transducers Let $\mathcal{X}$ be a finite set of variables denoted by $X, Y, \dots$ and B be a finite alphabet. A substitution σ is defined as a mapping $\sigma : \mathcal{X} \to (B \cup \mathcal{X})^*$. Let $\mathcal{S}_{\mathcal{X},B}$ be the set of all substitutions. Any substitution σ can be extended to $\hat{\sigma} : (B \cup \mathcal{X})^* \to (B \cup \mathcal{X})^*$ in a straightforward manner. The composition $\sigma_1\sigma_2$ of two substitutions σ_1 and σ_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 $u \in (B \cup \mathcal{X})^*$ is *k-linear* if each $X \in \mathcal{X}$ occurs at most k times in u. A substitution σ is k-linear if $\sigma(X)$ is *k-linear* for all X. It is *copyless* if for any variable X, there exists at most one variable Y such that X occurs in $\sigma(Y)$, and X occurs at most once in $\sigma(Y)$.

A *streaming string transducer* (SST) is a tuple $T = (A, B, Q, q_0, Q_f, \delta, \mathcal{X}, \rho, F)$ where (Q, q_0, Q_f, δ) is a one-way automaton, A and B are finite sets of input and output alphabets respectively, $\mathcal{X}$ is a finite set of variables, $\rho : \delta \to \mathcal{S}_{\mathcal{X},B}$ is a variable update and $F : Q_f \rightharpoonup (\mathcal{X} \cup B)^*$ is the output function.

Example 1. As an example, let $f : \{a, b\}^* \to \{a, b\}^*$ be the function mapping any word $u = a^{k_0}ba^{k_1} \cdots ba^{k_n}$ to the word $f(u) = a^{k_0}b^{k_0}a^{k_1}b^{k_1} \cdots a^{k_n}b^{k_n}$ obtained by adding after each block of consecutive a a block of consecutive b of the same

length. Since each word u over A can be uniquely written $u = a^{k_0}ba^{k_1}\cdots ba^{k_n}$ with some k_i being possibly equal to 0, the function f is well defined. We give in Figure 2 a 2DFT and an SST that realize f.

The concept of a run of an SST is defined in an analogous manner to that of a finite state automaton. The sequence $\langle\sigma_{r,i}\rangle_{0\leqslant i\leqslant|r|}$ of substitutions induced by a run $r = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \dots q_{n-1} \xrightarrow{a_n} q_n$ is defined inductively as the following: $\sigma_{r,i}{=}\sigma_{r,i-1}\rho(q_{i-1}, a_i)$ for $1 < i \leqslant |r|$ and $\sigma_{r,1} = \rho(q_0, a_1)$. We denote $\sigma_{r,|r|}$ by σ_r and say that σ_r is induced by r.

If r is accepting, i.e. $q_n \in Q_f$, we can extend the output function F to r by $F(r) = \sigma_\epsilon\sigma_r F(q_n)$, where σ_ϵ substitutes all variables by their initial value ϵ. For all words $u \in A^*$, the output of u by T is defined only if there exists an accepting run r of T on u, and in that case the output is denoted by $T(u) = F(r)$. The transformation $R(T)$ is then defined as the set of pairs $(u, T(u)) \in A^* \times B^*$.

An SST T is copyless if for every transition $t \in \delta$, the variable update $\rho(t)$ is copyless. Given an integer $k \in \mathbb{N}_{>0}$, we say that T is *k-bounded* if all its runs induce k-linear substitutions. It is *bounded* if it is k-bounded for some k.

The following theorem gives the expressiveness equivalence of the models we consider. We do not give the definitions of MSO graph transductions as our results will only involve state-based transducers (see [10] for more details).

Theorem 1 ([9, 1, 3]). *Let $f : A^* \to B^*$ be a function over words. Then the following conditions are equivalent:*

- *f is realized by an* MSO *graph transduction,*
- *f is realized by a* 2DFT*,*
- *f is realized by a copyless* SST*,*
- *f is realized by a bounded* SST*.*

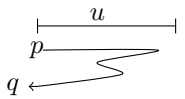
2.2 Transition monoid of transducers

A (finite) monoid M is a (finite) set equipped with an associative internal law $\cdot_M$ having a neutral element for this law. A morphism $\eta : M \to N$ between monoids is an application from M to N that preserves the internal laws, meaning that for all x and y in M, $\eta(x \cdot_M y) = \eta(x) \cdot_N \eta(y)$. When the context is clear, we will write xy instead of $x \cdot_M y$. A monoid M divides a monoid N if there exists an onto morphism from a submonoid of N to M. A monoid M is said to be *aperiodic* if there exists a least integer n, called the *aperiodicity index* of M, such that for all elements x of M, we have $x^n = x^{n+1}$.

Given an alphabet A, the set of words A^* is a monoid equipped with the concatenation law, having the empty word as neutral element. It is called the *free monoid* on A. A finite monoid M *recognizes* a language L of A^* if there exists an onto morphism $\eta : A^* \to M$ such that $L = \eta^{-1}(\eta(L))$. It is well-known that the languages recognized by finite monoids are exactly the regular languages.

The monoid we construct from a machine is called its *transition monoid*. We are interested here in aperiodic machines, in the sense that a machine is aperiodic if its transition monoid is aperiodic. We now give the definition of the transition monoid for a 2DFT and an SST.

Deterministic Two-Way Finite State Transducers As in the case of automata, the transition monoid of a 2DFT T is the set of all possible behaviors of T on a word. The following definition comes from [4], using ideas from [15] amongst others. As a word can be read in both ways, the possible runs are split into four relations over the set of states Q of T. Given an input word u, we define the left-to-left behavior $\mathrm{bh}_{\ell\ell}(u)$ as the set of pairs (p, q) of states of T such that there exists a run over u starting on the first letter of u in state p and exiting u on the left in state q (see Figure on the right). We define in an analogous fashion the left-to-right, right-to-left and right-to-right behaviors denoted respectively $\mathrm{bh}_{\ell r}(u)$, $\mathrm{bh}_{r\ell}(u)$ and $\mathrm{bh}_{rr}(u)$. Then the transition monoid of a 2DFT is defined as follows:



Let $T = (Q, A, \delta, q_0, F)$ be a 2DFT. The *transition monoid* of T is $A^*/\sim_T$ where $\sim_T$ is the conjunction of the four relations $\sim_{ll}$, $\sim_{lr}$, $\sim_{rl}$ and $\sim_{rr}$ defined for any words u, u' of A^* as follows: $u \sim_{xy} u'$ iff $\mathrm{bh}_{xy}(u) = \mathrm{bh}_{xy}(u')$, for $x, y \in \{\ell, r\}$. 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.

Note that since the set of states of T is finite, each behavior relation is of finite index and consequently the transition monoid of T is also finite. Let us also remark that the transition monoid of T does not depend on the output and is in fact the transition monoid of the underlying 2DFA.

Streaming String Transducers A notion of transition monoid for SST was defined in [12]. We give here its formal definition and refer to [12] for advanced considerations. In order to describe the behaviors of an SST, this monoid describes the possible flows of variables along a run. Since we give later an alternative definition of transition monoid for SST, we will call it the *flow transition monoid* (FTM).

Let T be an SST with states Q and variables $\mathcal{X}$. The *flow transition monoid* M_T of T is a set of square matrices over the integers enriched with a new absorbent element $\perp$. The matrices are indexed by elements of $Q \times \mathcal{X}$. Given an input word u, the image of u in M_T is the matrix m such that for all states p, q and all variables X, Y, $m[p, X][q, Y] = n \in \mathbb{N}$ (resp. $m[p, X][q, Y] = \perp$) if, and only if, there exists a run r of T over u from state p to state q, and X occurs n times in $\sigma_r(Y)$ (resp. iff there is no run of T over u from state p to state q).

Note that if T is k-bounded, then for all word w, all the coefficients of its image in M_T are bounded by k. The converse also holds. Then M_T is finite if, and only if, T is k-bounded, for some k.

It can be checked that the machines given in Example 1 are aperiodic. Theorem 1 extends to aperiodic subclasses and to first-order logic, as in the case of regular languages [14, 13]. These results as well as our contributions to these models are summed up in Figure 1.

Theorem 2 ([12, 4]). *Let $f : A^* \to B^*$ be a function over words. Then the following conditions are equivalent:*

- *f is realized by a* FO *graph transduction,*

– *f is realized by an aperiodic* 2DFT*,*
– *f is realized by an aperiodic* 1*-bounded* SST*.*

3 Substitution Transition Monoid

In this section, we give an alternative take on the definition of the transition monoid of an SST, and show that both notions coincide on aperiodicity and boundedness. The intuition for this monoid, that we call the *substitution transition monoid*, is for the elements to take into account not only the multiplicity of the output of each variable in a given run, but also the order in which they appear in the output. It can be seen as an enrichment of the classic view of transition monoids as the set of functions over states equipped with the law of composition. Given a substitution $\sigma \in \mathcal{S}_{\mathcal{X},B}$, let us denote $\tilde{\sigma}$ the projection of σ on the set $\mathcal{X}$, i.e. we forget the parts from B. The substitutions $\tilde{\sigma}$ are homomorphisms of $\mathcal{X}^*$ which form an (infinite) monoid. Note that in the case of a 1-bounded SST, each variable occurs at most once in $\tilde{\sigma}(Y)$.

Substitution Transition Monoid of an SST*.* Let T be an SST with states Q and variables $\mathcal{X}$. The *substitution transition monoid* (STM) of T, denoted M_T^σ, is a set of partial functions $f : Q \rightharpoonup Q \times \mathcal{S}_{\mathcal{X},\emptyset}$. Given an input word u, the image of u in M_T^σ is the function f_u such that for all states p, $f_u(p) = (q, \tilde{\sigma}_r)$ if, and only if, there exists a run r of T over u from state p to state q that induces the substitution $\tilde{\sigma}_r$. This set forms a monoid when equipped with the following composition law: Given two functions $f_u, f_v \in M_T^\sigma$, the function f_{uv} is defined by $f_{uv}(q) = (q'', \tilde{\sigma} \circ \tilde{\sigma}')$ whenever $f_u(q) = (q', \tilde{\sigma})$ and $f_v(q') = (q'', \tilde{\sigma}')$.

We now make a few remarks about this monoid. Let us first observe that the FTM of T can be recovered from its STM. Indeed, the matrix m associated with a word u in M_T is easily deduced from the function f_u in M_T^σ. This observation induces an onto morphism from M_T^σ to M_T, and consequently the FTM of an SST divides its STM. This proves that if the STM is aperiodic, then so is the FTM since aperiodicity is preserved by division of monoids. Similarly, copyless and k-bounded SST (given $k \in \mathbb{N}_{>0}$) are characterized by means of their STM. This transition monoid can be separated into two main components: the first one being the transition monoid of the underlying deterministic one-way automaton, which can be seen as a set of functions $Q \to Q$, while the second one is the monoid $\mathcal{S}_\mathcal{X}$ of homomorphisms on $\mathcal{X}$, equipped with the composition. The aware reader could notice that the STM can be written as the wreath product of the transformation semigroup $(\mathcal{X}^*, \mathcal{S}_\mathcal{X})$ by (Q, Q^Q). However, as the monoid of substitution is obtained through the closure under composition of the homomorphisms of a given SST, it may be infinite.

The next theorem proves that aperiodicity for both notions coincide, since the converse comes from the division of STM by FTM.

Theorem 3. *Let T be a k-bounded SST with ℓ variables. If its* FTM *is aperiodic with aperiodicity index n then its* STM *is aperiodic with aperiodicity index at most $n + (k+1)\ell$.*

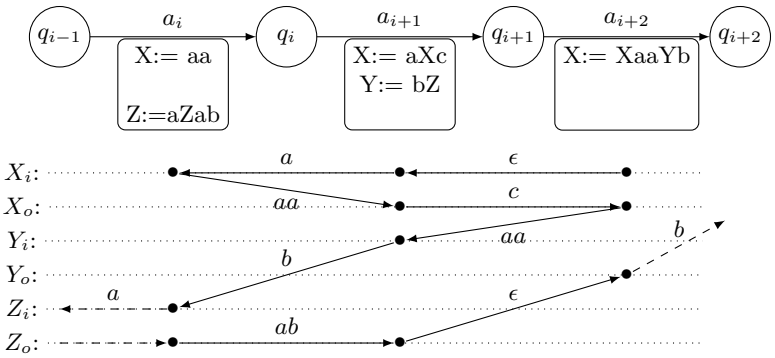


Fig. 3: The output structure of a partial run of an SST used in the proof of Theorem 4.

4 From 1-bounded SST to 2DFT

The existing transformation of a 1-bounded (or even copyless) SST into an equivalent 2DFT goes through MSO transductions, yielding a non-elementary complexity. We present here an original construction whose complexity is elementary.

Theorem 4. *Let T be a 1-bounded SST with n states and m variables. Then we can effectively construct a deterministic 2-way transducer that realizes the same function. If T is 1-bounded (resp. copyless), then the* 2DFT *has $O(m2^{m2^m}n^n)$ states (resp. $O(mn^n)$).*

Proof. We define the 2DFT as the composition of a left-to-right sequential transducer, a right-to-left sequential transducer and a 2-way transducer. Remark that this proves the result as two-way transducers are closed under composition with sequential ones [5].

The left-to-right sequential transducer does a single pass on the input word and outputs the same word enriched with the transition used by the SST in the previous step. The right-to-left transducer uses this information to enrich each position of the input word with the set of useful variables, i.e the variables that flow to an output variable according to the partial run on the suffix read. The two sequential transducers are quite standard. They realize length-preserving functions that simply enrich the input word with new information. The last transducer is more interesting: it uses the enriched information to follow the output structure of T. The *output structure* of a run is a labeled and directed graph such that, for each variable X useful at a position j, we have two nodes X_i^j and X_o^j linked by a path whose concatenated labels form the value stored in X at position j of the run (see [12] and Figure 3).

The transition function of the two-way transducer is described in Figure 4. It first reaches the end of the word and picks the first variable to output. It then rewinds the run using the information stored by the first sequential transducer, producing the said variable using the local update function. When it has finished

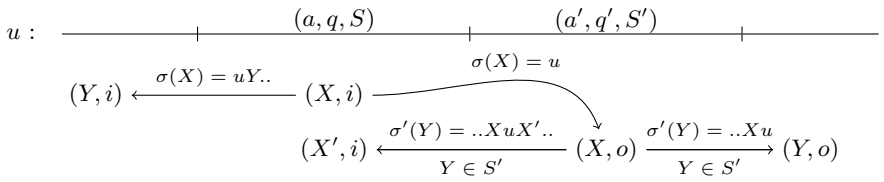


Fig. 4: The third transducer follows the output structure. States indexed by i correspond to the beginning of a variable, while states indexed by o correspond the end. σ (resp. σ') stand for the substitution at position a (resp. a').

to compute and produce a variable X, it switches to the following one using the information of the second transducer to know which variable Y X is flowing to, and starts producing it. Note that such a Y is unique thanks to the 1-boundedness property. If T is copyless, then this information is local and the second transducer can be bypassed.

Regarding complexity, a careful analysis of the composition of a one-way transducer of size n with a two-way transducer of size m from [6, 4] shows that this can be done by a two-way transducer of size $O(mn^n)$. Then given a 1-bounded SST with n states and m variables, we can construct a deterministic two-way transducer of size $O(m2^{m2^m}n^n)$. If T is copyless, the second sequential transducer is omitted, resulting in a size of $O(mn^n)$.

Theorem 5. *Let T be an aperiodic 1-bounded SST. Then the equivalent* 2DFT *constructed using Theorem 4 is also aperiodic.*

Proof. The aperiodicity of the three transducers gives the result as aperiodicity is preserved by composition of a one-way by a two-way [4]. The aperiodicity of the two sequential transducers is straightforward since their runs depend respectively on the underlying automaton and the update function. The aperiodicity of the 2DFT comes from the fact that since it follows the output structure of the SST, its partial runs are induced by the flow of variables and their order in the substitutions, which is an information contained in the FTM and thus aperiodic thanks to Theorem 3.

5 From 2DFT to copyless SST

In [1], the authors give a procedure to construct a copyless SST from a 2DFT. This procedure uses the intermediate model of heap based transducers. We give here a direct construction with similar complexity. This simplified presentation allows us tu prove that the construction preserves the aperiodicity.

Theorem 6. *Let T be a* 2DFT *with n states. Then we can effectively construct a copyless SST with $O((2n)^{2n})$ states and $2n-1$ variables that computes the same function.*

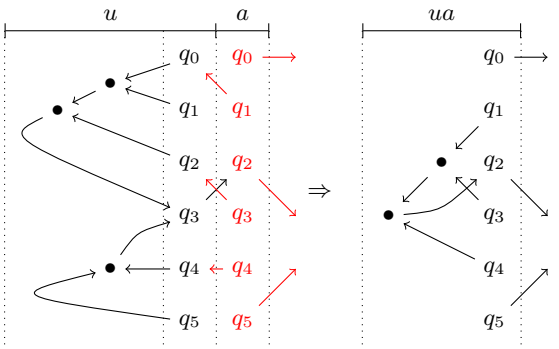


Fig. 5: Left: The state of the SST is represented in black. The red part corresponds to the local transitions of the 2DFT. Right: After reading a, we reduce the new forest by eliminating the useless branches and shortening the unlabeled linear paths.

Proof. (Sketch of) The main idea is for the constructed SST to keep track of the right-to-right behavior of the prefix read until the current position, similarly to the construction of Shepherdson [15]. This information can be updated upon reading a new letter, constructing a one-way machine recognizing the same input language. The idea from [3] is to have one variable per possible right-to-right run, which is bounded by the number of states. However, since two right-to-right runs from different starting states can merge, this construction results in a 1-bounded SST. To obtain copylessness, we keep track of these merges and the order in which they appear. Different variables are used to store the production of each run before the merge, and one more variable stores the production after.

The states of the copyless SST are represented by sets of labeled trees having the states of the input 2DFT as leaves. Each inner vertex represents one merging, and two leaves have a common ancestor if the right-to-right runs from the corresponding states merge at some point. Each tree then models a set of right-to-right runs that all end in a same state. Note that it is necessary to also store the end state of these runs. For each vertex, we use one variable to store the production of the partial run corresponding to the outgoing edge.

Given such a state and an input letter, the transition function can be defined by adding to the set of trees the local transitions at the given letter, and then reducing the resulting graph in a proper way (see Figure 5).

Finally, as merges occur upon two disjoint sets of states of the 2DFT (initially singletons), the number of merges, and consequently the number of inner vertices of our states, is bounded by $n-1$. Therefore, an input 2DFT with n states can be realized by an SST having $2n-1$ variables. Finally, as states are labeled graphs, Cayley's formula yields an exponential bound on the number of states.

Moreover, this construction preserves aperiodicity:

Theorem 7. *Let T be an aperiodic* 2DFT. *Then the equivalent* SST *constructed using Theorem 6 is also aperiodic.*

Proof. If the input 2DFT is aperiodic of index n, then for any word w, w^n and w^{n+1} merge the same partial runs for the four kinds of behaviors, by definition, and in fact the merges appear in the same order. As explained earlier, the state q_1 (resp. q_2) reached by the constructed SST over the inputs uw^n (resp. uw^{n+1}) represents the merges of the right-to-right runs of T over uw^n (resp. uw^{n+1}).

Since these runs can be decomposed in right-to right runs over u and partial runs over w^n and w^{n+1}, the merge equivalence between w^n and w^{n+1} implies that $q_1 = q_2$. Moreover, since variables are linked to these merges, the aperiodicity of the merge equivalence implies the aperiodicity of both the underlying automaton and the substitution function of the SST, concluding the proof.

As a corollary, we obtain that the class of aperiodic copyless SST is expressively equivalent to first-order definable string-to-string transductions.

Corollary 1. *Let $f : A^* \to B^*$ be a function over words. Then f is realized by a* FO *graph transduction iff it is realized by an aperiodic copyless* SST.

6 From k-bounded to 1-bounded SST

The existing construction from k-bounded to 1-bounded, presented in [3], builds a copyless SST. We present an alternative construction that, given a k-bounded SST, directly builds an equivalent 1-bounded SST. We will prove that this construction preserves aperiodicity.

Theorem 8. *Given a k-bounded* SST *T with n states and m variables, we can effectively construct an equivalent 1-bounded* SST*. This new* SST *has $n2^N$ states and mkN variables, where $N = O(n^n(k+1)^{nm^2})$ is the size of the flow transition monoid M_T.*

Proof. In order to move from a k-bounded SST to a 1-bounded SST, the natural idea is to use copies of each variable. However, we cannot maintain k copies of each variable all the time: suppose that X flows into Y and Z, which both occur in the final output. If we have k copies of X, we cannot produce in a 1-bounded way (and we do not need to) k copies of Y and k copies of Z.

Now, if we have access to a look-ahead information, we can guess how many copies of each variable are needed, and we can easily construct a copyless SST by using exactly the right number of copies for each variable and at each step. The construction relies on this observation. We simulate a look-ahead through a subset construction, having copies of each variable for each possible behavior of the suffix. Then given a variable and the behavior of a suffix, we can maintain the exact number of variables needed and perform a copyless substitution to a potential suffix for the next step. However, since the SST is not necessarily co-deterministic, a given suffix can have multiple successors, and the result is that its variables flow to variables of different suffixes. As variables of different suffixes are never recombined, we obtain a 1-bounded SST.

Theorem 9. *Let T be an aperiodic k-bounded* SST*. Then the equivalent 1-bounded* SST *constructed using Theorem 8 is also aperiodic.*

As a corollary, we obtain that for the class of aperiodic bounded SST is expressively equivalent to first-order definable string-to-string transductions.

Corollary 2. *Let $f : A^* \to B^*$ be a function over words. Then f is realized by a* FO *graph transduction iff it is realized by an aperiodic bounded* SST *($k \in \mathbb{N}_{>0}$).*

7 Perspectives

There is still one model equivalent to the generic machines whose aperiodic subclass elude our scope yet, namely the *functional two-way transducers*, which correspond to non-deterministic two-way transducers realizing a function. To complete the picture, a natural approach would then be to consider the constructions from [8] and prove that aperiodicity is preserved. One could also think of applying this approach to other varieties of monoids, such as the $\mathcal{J}$-trivial monoids, equivalent to the boolean closure of existential first-order formulas $\mathcal{B}\Sigma_1[<]$. Unfortunately, the closure of such transducers under composition requires some strong properties on varieties (at least closure under semidirect product) which are not satisfied by varieties less expressive than the aperiodic. Consequently the construction from SST to 2DFT cannot be applied. On the other hand, the other construction could apply, providing one inclusion. Then an interesting question would be to know where the corresponding fragment of logic would position.

References

1. R. Alur and P. Černý. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs.*, pages 1–12. Schloss Dagstuhl. Leibniz-Zent. Inform., 2010.
2. R. Alur, A. Durand-Gasselin, and A. Trivedi. From monadic second-order definable string transformations to transducers. In *LICS*, pages 458–467, 2013.
3. R. Alur, E. Filiot, and A. Trivedi. Regular transformations of infinite strings. In *LICS*, pages 65–74, 2012.
4. O. Carton and L. Dartois. Aperiodic two-way transducers and fo-transductions. In *CSL*, volume 41 of *LIPIcs*, pages 160–174. Schloss Dagstuhl. Leibniz-Zent. Inform., 2015.
5. M. P. Chytil and V. Jákl. Serial composition of 2-way finite-state transducers and simple programs on strings. In *Automata, languages and programming*, pages 135–137. LNCS, Vol. 52. Springer, Berlin, 1977.
6. L. Dartois. *Méthodes algébriques pour la théorie des automates.* PhD thesis, LIAFA-Université Paris Diderot, Paris, 2014.
7. L. Dartois, I. Jecker, and P.-A. Reynier. Aperiodic string transducers. *CoRR*, abs/1506.04059, 2015.
8. R. de Souza. Uniformisation of two-way transducers. In *LATA*, pages 547–558, 2013.
9. J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.
10. E. Filiot. Logic-automata connections for transformations. In *ICLA*, volume 8923 of *LNCS*, pages 30–57. Springer, 2015.
11. E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. From two-way to one-way finite state transducers. In *LICS*, pages 468–477. IEEE Computer Society, 2013.
12. E. Filiot, S. N. Krishna, and A. Trivedi. First-order definable string transformations. In *FSTTCS*, volume 29 of *LIPIcs*, pages 147–159. Schloss Dagstuhl - Leibniz-Zent. Inform., 2014.
13. R. McNaughton and S. Papert. *Counter-free automata.* The M.I.T. Press, Cambridge, Mass.-London, 1971.

14. M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.
15. J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.