Aperiodic String Transducers *

Luc Dartois, Ismaël Jecker

Université Libre de Bruxelles, Belgium

Pierre-Alain Reynier

Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

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) [8], 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 [8, 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 trans-

*This work is supported by the ARC project Transform (French speaking community of Belgium), the Belgian FNRS PDR project Flare, the PHC project VAST (35961QJ) funded by Campus France and WBI, and the DeLTA project (ANR-16-CE40-0007).

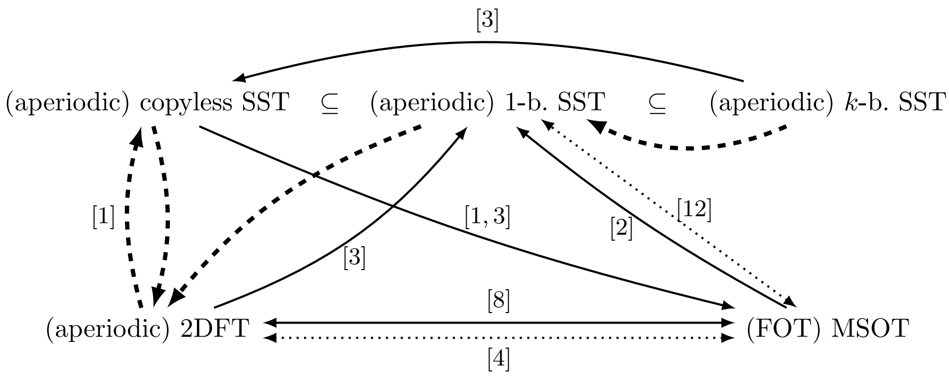


Fig. 1: Summary of transformations between equivalent models. k-b. stands for k-bounded. Plain (resp. dotted) arrows concern regular models (resp. bracketed models). Original constructions presented in this paper are depicted by thick dashed arrows and are valid for both regular and aperiodic versions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

ductions. 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, the class of rational functions (accepted by one-way finite state transducers) owns a simple characterization in terms of logic, as shown in [9]. 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]. 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, although some recent work was done on one-way transducers [10].

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).

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), stays at the same

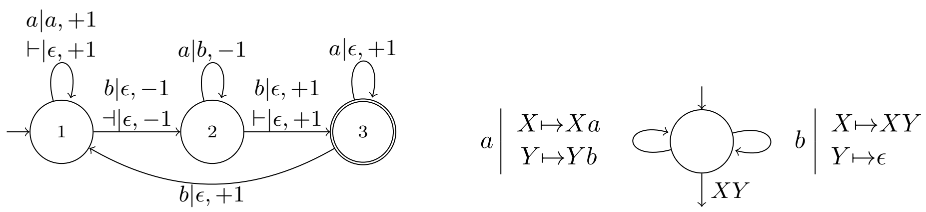


Fig. 2: Aperiodic 2DFT (left) and SST (right) realizing the function f.

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* r of $\mathcal{A}$ is a finite sequence of configurations. The run $r = (p_1, i_1) \dots (p_m, i_m)$ is a run on an input word $u \in A^*$ of length n if $i_m \leqslant n+1$, and for all $k \in \{1, \dots, m-1\}$, $0 \leqslant i_k \leqslant n+1$ and $\delta(p_k, u[i_k]) = (p_{k+1}, i_{k+1} - i_k)$. 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) from A to B extend 2DFA with a one-way left-to-right output tape. More formally, the transition relation is extended with outputs: $\delta : Q \times (A \uplus \{\vdash, \dashv\}) \to B^* \times Q \times \{-1, 0, +1\}$. When a transition with right-hand side $(v, q, m) \in B^* \times Q \times \{-1, 0, +1\}$ is fired, the word v is appended to the right of the output tape. Formally, 2DFT are defined as tuples $T = (A, B, Q, q_0, F, \delta)$.

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 r 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 r. Given a 2DFT T, an input word $u \in A^*$ and a run $r = (p_1, i_1) \dots (p_m, i_m)$ of T on u, the output of r on u is the word obtained by concatenating the outputs of the transitions followed by r. If r 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 r 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 deterministic one-way automaton over the alphabet A, B is the finite output alphabet, $\mathcal{X}$ is the finite set of variables, $\rho : Q \times (A \uplus \{\vdash, \dashv\}) \to \mathcal{S}_{\mathcal{X},B}$ is the variable update function and $F : Q_f \rightharpoonup (\mathcal{X} \cup B)^*$ is the output function.

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 $(u, T(u)) \in A^* \times B^*$.

Example 1. *As an example, let $f : \{a, b\}^* \to \{a, b\}^*$ be the function mapping any word $u = a^{k_0} b a^{k_1} \cdots b a^{k_n}$ 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 $u = a^{k_0} b a^{k_1} \cdots b a^{k_n}$ 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.*

An SST T is copyless if for every $(q, a) \in Q \times (A \uplus \{\vdash, \dashv\})$, the variable update $\rho(q, a)$ 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. Since our results will only involve state-based transducers, we give no definition of MSO graph transductions (see [9] for more details).

Theorem 2 ([8, 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*.*

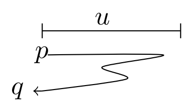
2.2. *Transition monoid of transducers*

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 on 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 $T = (A, B, Q, q_0, F, \delta)$ be a 2DFT. The *transition monoid* of T is $(A \uplus \{\vdash, \dashv\})^*/{\sim_T}$ 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 on u from state p to state q, and X occurs n times in $\sigma_r(Y)$ (resp. iff there is no run of T on u from state p to state q).

Note that if T is k-bounded, then for every 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 2 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 3 ([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 $\tilde{\sigma}(Y)$, and in the case of a copyless SST, the monoid is finite.

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 $f : Q \rightharpoonup Q \times \mathcal{S}_{\mathcal{X}}$, where $\mathcal{S}_{\mathcal{X}}$ is the monoid of homomorphisms on $\mathcal{X}$, equipped with the composition. 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 on 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 $\mathcal{S}_\mathcal{X}$. The aware reader could notice that the STM divides the wreath product of transformation semigroups $(Q, Q^Q) \circ (\mathcal{X}^*, \mathcal{S}_\mathcal{X})$. However, as the monoid of substitutions is obtained through the closure under composition of the homomorphisms of a given SST, it may be infinite while the STM is always finite for k-bounded SST.

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

Theorem 4. *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 $n + (k+1)\ell$.*

Proof. Let T be a k-bounded SST. We define a *loop* as the run induced by a pair $(q, u) \in Q \times A^*$ such that $\delta(q, u) = q$. Suppose now that M_T is aperiodic, and let n be its aperiodicity index. Wlog, we assume that the transition function of T is complete. This implies that for all states p of T, there exists a state q such that $p \xrightarrow{u^n} q \xrightarrow{u} q$. Then if the image of the loops (i.e. the set of all $\tilde{\sigma}$ such that there exists a loop (q, u) where $f_u(q) = (q, \tilde{\sigma})$) in the STM is aperiodic with index m, then the STM is aperiodic with index at most $n + m$.

Consequently, in the following σ denotes the substitution of a loop of T, and we aim to prove that $\tilde{\sigma}^{(k+1)\ell} = \tilde{\sigma}^{(k+1)\ell+1}$.

Before proving this though, we define the relation $\lessdot \subseteq \mathcal{X} \times \mathcal{X}$ as follows. Given two variables X and Y, we have $X \lessdot Y$ if there exists a positive integer i such that X flows into Y in σ^i. This relation is clearly transitive. The next lemma proves that it is also anti-symmetric, hence we can use this relation as an induction order to prove the result.

Lemma 5. *Given two different variables X and Y, if $X \lessdot Y$, then $Y \not\lessdot X$.*

Proof. We proceed by contradiction and assume that there exist two different variables X and Y and two integers i and j such that X occurs in $\sigma^i(Y)$ and Y occurs in $\sigma^j(X)$.

Then for any $m > 0$, X occurs in $\sigma^{m(i+j)}(X)$ and Y occurs in $\sigma^{m(i+j)+j}(X)$. As T is aperiodic of index n, for m large enough we have $\sigma^{m(i+j)} = \sigma^{m(i+j)+j} = \sigma^n$ and thus both X and Y occur in both $\sigma^n(X)$ and $\sigma^n(Y)$. Then $\sigma^{2n}(X)$ contains both $\sigma^n(X)$ and $\sigma^n(Y)$ and thus contains at least two occurrences of X and Y. By aperiodicity we have $\sigma^{2n}(X) = \sigma^n(X)$ thus $\sigma^n(X)$ contains two occurrences of X. By iterating this process, we prove that the number of occurrences of X in $\sigma^n(X)$ is not bounded, yielding a contradiction. □

We now prove that for all variables X in $\mathcal{X}$, $\tilde{\sigma}^{(k+1)\ell}(X) = \tilde{\sigma}^{(k+1)\ell+1}(X)$ by treating the following two cases:

- If $X \in \sigma(X)$, then either $\tilde{\sigma}(X) = X$ and then $\tilde{\sigma}^2(X) = \tilde{\sigma}(X)$, or there exists $Y \neq X$ such that $Y \in \tilde{\sigma}(X)$. In the latter case, we get by iteration that for all $i > 0$, $|\tilde{\sigma}^i(X)| > \Sigma_{j<i}|\tilde{\sigma}^j(Y)|$. Then as T is k-bounded, we have $|\tilde{\sigma}^i(X)| \leqslant k\ell$ and thus $\Sigma_{j<i}|\tilde{\sigma}^j(Y)|$ is bounded, and $\tilde{\sigma}^{k\ell}(Y) = \epsilon$, which proves that $\tilde{\sigma}^{k\ell}(X) = \tilde{\sigma}^{k\ell+1}(X)$.
- If $X \notin \sigma(X)$, let us consider the relation $\lessdot$. By Lemma 5 this relation is cycle-free. Then there is a lesser level, on which there are the variables Y such that $\tilde{\sigma}(Y) \subseteq \{Y\}$. There, either $\tilde{\sigma}(Y) = \emptyset$ and aperiodicity becomes trivial, or $\tilde{\sigma}(Y) = Y$ and the case was dealt with in the previous point and is thus aperiodic with index $k\ell$. Now we can end the proof by reasoning by induction on $\lessdot$, as if $X \notin \sigma(X)$ and all variables $Y \lessdot X$ are aperiodic with index i, then $\tilde{\sigma}^{i+1}(X)$ can be written as the concatenation of $\tilde{\sigma}^i(Y)$, for aperiodic variables Y of index i. Then $\tilde{\sigma}^{i+1}(X) = \tilde{\sigma}^{i+2}(X)$. We conclude by noticing that the length of the longest chain of $\lessdot$ is bounded by ℓ.

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 6. *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. Let $T = (A, B, Q, q_0, Q_f, \delta, \mathcal{X}, \rho, F)$ be an SST. Let us construct a two-way transducer $\mathcal{A}$ that realizes the same function. The transducer $\mathcal{A}$ will follow the output structure (see Figure 3) of T and construct the output as it appears in the structure. To make the proof easier to read, we define $\mathcal{A}$ as the composition of a

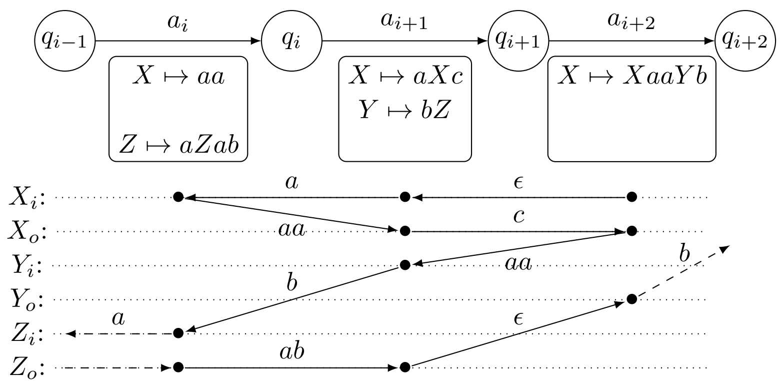


Fig. 3: The output structure of a partial run of an SST used in the proof of Theorem 6.

left-to-right sequential transducer $\mathcal{A}'$, a right-to-left sequential transducer $\mathcal{A}''$ and a 2-way transducer $\mathcal{B}$. Remark that this proves the result as two-way transducers are closed under composition with sequential ones [5]. The transducer $\mathcal{A}'$ does a single pass on the input and enriches it with the transition used by T in the previous step. The second transducer uses this information, and enriches the input word with the set of variables corresponding to the variables that will be produced from this position. 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). Formally, the output structure of a run $q_0 \xrightarrow{u} q_n$ is the oriented graph over $\mathcal{X} \times [1, |u|] \times \{i, o\}$ whose edges are labeled by output words and are of the form:

- $((X, j, i), v, (Y, j-1, i))$ if $\rho(q_{j-1}, a_j, X)$ starts with vY.
- $((X, j, o), v, (Y, j-1, i))$ if there exists Z such that XvY appears in $\rho(q_{j-1}, a_j, Z)$,
- $((X, j, o), v, (Y, j+1, o))$ if $\rho(q_{j-1}, a_j, Y)$ ends by Xv,
- $((X, j, i), v, (X, j+1, o))$ if $\rho(q_{j-1}, a_j, X) = v$.

We furthermore restrict to the connected component corresponding to the actual output of the run.

The set of variables (added to the input word by $\mathcal{A}''$) will be used to clear the nondeterminism due to the 1-bounded property. Note that in the case of a copyless SST, the transducer $\mathcal{A}''$ can be omitted. We now explain how the several transducers behave on a given run $r = q_0 \xrightarrow{a_1} q_1 \ldots \xrightarrow{a_n} q_n$. To ease the notations,

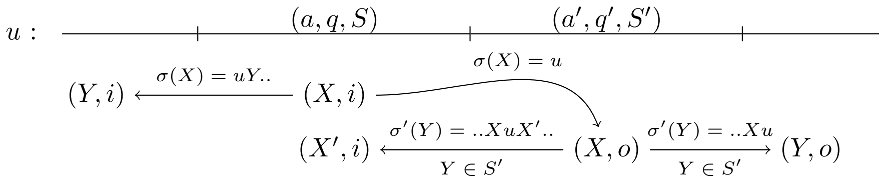


Fig. 4: The third transducer follows the output structure. States indexed by i correspond to the beginning of a variable, while states indexed by o correspond to the end. σ (resp. σ') stands for the substitution at position a (resp. a').

we let $A_{\dashv} = A \uplus \{\dashv\}$.

The transducer $\mathcal{A}' = (A, A_{\dashv} \times Q, Q \uplus \{i', f'\}, \{i'\}, \{f'\}, \delta')$, which enriches the input word with the transitions of the previous step, can be done easily with a 1-way transducer. The transitions of δ' are defined as follows: first, we have $i' \xrightarrow{\vdash|\epsilon} q_0$. Second, we have $q \xrightarrow{\sigma|(\sigma,q)} q'$ for all $q \in Q$ and $\sigma \in A_{\dashv}$, where $q' = \delta(q, a)$ if $\sigma \in A$, and $q' = f$ otherwise. Note that one can recover the transition taken by the SST from the letter read and the previous state. Then on the run r, if $u = a_1 \dots a_n$, we get the output word $\mathcal{A}'(\vdash u \dashv) = (a_1, q_0)(a_2, q_1) \dots (a_n, q_{n-1})(\dashv, q_n)$.

The transducer $\mathcal{A}'' = (A_{\dashv} \times Q, A_{\dashv} \times Q \times 2^{\mathcal{X}}, 2^{\mathcal{X}} \uplus \{i'', f''\}, \{i''\}, \{f''\}, \delta'')$, which enriches each letter of the input word (except the endmarkers) with the set of variables involved in the final output from this step, can be done with a right-to-left sequential transducer. The last symbol $(\dashv, q_n)$ allows to identify the set of variables appearing in $F(q_n)$. This information can be propagated while getting back in the run of the SST. Formally, the transitions of δ'' are defined as follows:

$$\begin{cases} i'' \xrightarrow{\dashv|\epsilon} i'' \\ i'' \xrightarrow{(\dashv,q)|(\dashv,q,\emptyset)} \{X \in \mathcal{X} \mid X \text{ appears in } F(q)\} \\ S \xrightarrow{(a,q)|(a,q,S)} \{X \in \mathcal{X} \mid \exists Y \in S,\ X \text{ occurs in } \rho(q, a, Y)\} \quad \text{for all } q \in Q, a \in A \\ S \xrightarrow{\vdash|\epsilon} f'' \end{cases}$$

Given the run r of the SST on the input word u, we define $S_n = F(q_n)$, and $S_{i-1} = \{X \in \mathcal{X} \mid \exists Y \in F(q_n)\ X \in \sigma_{q_{i-1}, a_i \dots a_n}(Y)\}$. Then we get the output word $\mathcal{A}'' \circ \mathcal{A}'(\vdash u \dashv) = (a_1, q_0, S_1)(a_2, q_1, S_2) \dots (a_n, q_{n-1}, S_n)(\dashv, q_n, \emptyset)$.

The aim of the third transducer $\mathcal{B}$ is to follow the output structure of T. Let $\mathcal{B} = (A_{\dashv} \times Q \times 2^{\mathcal{X}}, B, P, \{p_0\}, \{p_f\}, \delta_{\mathcal{B}})$ be defined as follows. First, $P = (\mathcal{X} \times \{i, o\}) \uplus \{p_0, p_f\}$ is the set of states. The transducer does a first left-to-right reading of the input in state p_0. The subset $\mathcal{X} \times \{i, o\}$ will then be used to follow the output structure while keeping track of which variable we are currently producing. The set $\{i, o\}$ stands for *in* and *out* and corresponds to the similar notions in the output structure. Informally, *in* states will move to the left, while *out* states move to the right. The states p_0 and p_f are new states that are respectively initial and final.

The transition function $\delta_{\mathcal{B}}$ is detailed below (see also Figure 4). In the following, we consider that the transducer is in state p reading the triplet $t = (a, q, S) \in A_{\dashv} \times Q \times 2^{\mathcal{X}}$, or one of the endmarkers:

- If $p = p_0$ and $a \neq \dashv$, then we set $\delta_{\mathcal{B}}(p_0, t) = (\epsilon, p_0, +1)$.
- If $p = p_0$ and $a = \dashv$, then if $F(q)$ starts by uX with $u \in B^*$ and $X \in \mathcal{X}$, then $\delta_{\mathcal{B}}(p, t) = (u, (X, i), -1)$.
- If $p = (X, i)$, and $t \neq \vdash$ then:
  - either $\rho(q, a)(X) = u \in B^*$ and does not contain any variable, and we set $\delta_{\mathcal{B}}(p, t) = (u, (X, o), +1)$,
  - or $\rho(q, a)(X)$ starts by uY with $u \in B^*$ and $Y \in \mathcal{X}$, then $\delta_{\mathcal{B}}(p, t) = (u, (Y, i), -1)$.
- If $p = (X, i)$, and $t = \vdash$ then $\delta_{\mathcal{B}}(p, t) = (\epsilon, (X, o), +1)$.
- If $p = (X, o)$ and $a \neq \dashv$, then let Y be the unique variable of S such that X appears in $\rho(q, a)(Y)$. Then we have:
  - either $\rho(q, a)(Y)$ ends by Xu with u in B^* and we set $\delta_{\mathcal{B}}(p, t) = (u, (Y, o), +1)$,
  - or $\rho(q, a)(Y)$ is of the form $(B \cup \mathcal{X})^* XuX'(B \cup \mathcal{X})^*$ and we set $\delta_{\mathcal{B}}(p, t) = (u, (X', i), -1)$.

  Note that the unicity of such Y in S is due to the 1-boundedness property. If T is copyless, then this information is irrelevant and $\mathcal{A}''$ can be bypassed.
- If $p = (X, o)$, $q \in Q_f$ and $a = \dashv$ then:
  - either $F(q)$ ends by Xu with u in B^* and we set $\delta_{\mathcal{B}}(p, t) = (u, p_f, +1)$,
  - or $F(q)$ is of the form $(B \cup \mathcal{X})^* XuX'(B \cup \mathcal{X})^*$ and we set $\delta_{\mathcal{B}}(p, t) = (u, (X', i), -1)$.

Then we can conclude the proof as $T = \mathcal{B} \circ \mathcal{A}'' \circ \mathcal{A}'$ and 2-way transducers are closed under composition [5].

Regarding complexity, a careful analysis of the composition of a one-way transducer of size m with a two-way transducer of size n from [6, 4] shows that this can be done by a two-way transducer of size $O(nm^m)$. Then given a 1-bounded SST with n states and m variables, we can construct a deterministic two-way transducer of size $O(m(2^m)^{2^m} n^n) = O(m2^{m2^m} n^n)$. If T is copyless, the sequential right-to-left transducer can be omitted, and the resulting 2DFT is of size $O(mn^n)$. □

Theorem 7. *Let T be an aperiodic 1-bounded SST. Then the equivalent* 2DFT *constructed using Theorem 6 is also aperiodic.*

Proof. We prove separately the aperiodicity of the three transducers. Then the result comes from the fact that aperiodicity is preserved by composition of a one-way by a two-way [4].

First, consider the transducer $\mathcal{A}'$. It is a one-way transducer that simply enriches the input word with transitions from T, each enrichment corresponding to the transition taken by T in the previous step. Then since T is aperiodic, so is its underlying automaton. Then the enrichment and thus $\mathcal{A}'$ are aperiodic.

Secondly, given an input word, the transducer $\mathcal{A}''$ stores at each position the set of variables that will be output by T. Now as T is aperiodic, the flow of variable is aperiodic. Thus the value taken by this set is aperiodic and so is $\mathcal{A}''$.

Now, consider the transducer $\mathcal{B}$ and a run r of $\mathcal{B}$ on u^n starting in state p. Note that the fact that there exists a run on an enriched input word v implies that it is well founded, meaning that it is the image of some word of A^* by $\mathcal{A}'' \circ \mathcal{A}'$. If p is of the form (X, i), then the run starts from the right of u^n and follows the substitution $\sigma_r(X)$. It exits u^n either in state (X, o) on the right if $\sigma_r(X)$ is a word of B^*, or in a state (Y, i) on the left where Y is the first variable appearing in $\sigma_r(X)$. In both cases the state at the end of the run only depends on the underlying automata of T and the order of variables appearing in the substitution induced by the run. Since the substitution transition monoid is aperiodic if T is aperiodic by Theorem 4, a similar run exists on u^{n+1}.

Finally, if p is of the form (X, o), then the state in which the run exits u^n depends on the unique variable Y such that X belongs to $\sigma_r(Y)$ and Y belongs to the set of variables of the last letter of the input. Then the run follows the substitution $\sigma_r(Y)$. It will exit the input word in state (X', i) on the left if X' appears in $\tilde{\sigma}_r(Y)$ for some variable X' and in state (Y, o) otherwise. As the flow of variable as well as the underlying automaton are aperiodic, a similar run exists on u^{n+1}.

We conclude the proof by noticing that the same arguments will hold to reduce runs on u^{n+1} to runs on u^n. □

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 to prove that the construction preserves the aperiodicity.

Theorem 8. *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.*

The main idea is for the constructed SST T' 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

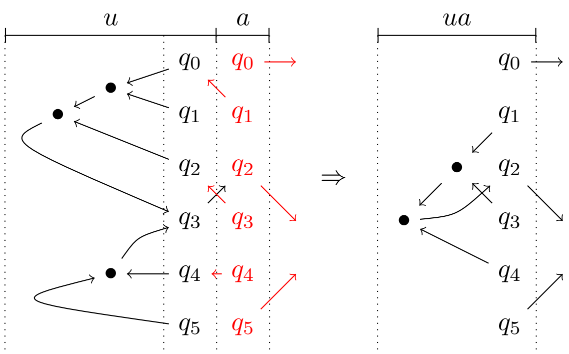


Fig. 5: Left: The state of the SST is represented in black. The red part corresponds to the local transitions of the 2DFT. Right: After reading a, we reduce the new forest by eliminating the useless branches and shortening the unlabeled linear paths.

more variable stores the production after.

The states of T' 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.

The remainder of the section is dedicated to give a formal proof of this intuition. Subsection 5.1 describes the merging forests that are the states of the SST, and Subsection 5.2 proves complexity bounds on the size of these objects. These bounds are then used in Subsection 5.3 to precisely define the substitution function. The formal construction is presented in Subsection 5.4 and its aperiodicity proved in Subsection 5.5.

5.1. *Merging forests*

Let $T = (A, B, Q, q_0, F, \delta)$ be a 2DFT. We suppose that the transducer T starts to read its input from the end, and not from the beginning, i.e., given an input w, the initial configuration is $(q_0, |w|+1)$. Moreover, we also assume that T produces no output while reading the symbol $\vdash$, i.e., for any every $p \in Q$, $\delta(p, \vdash) = (\epsilon, q, +1)$ for some $q \in Q$. Note that any 2DFT can be transformed with ease into a transducer satisfying those two properties.

For every word w over the alphabet $\bar{A}^*$, we expose a pair (G_w, ϕ_w) that contains the information concerning the right-to-right runs of T on the input word $\vdash w$, and their mergings. It is composed of a forest G_w whose branches represent right-to-right runs and vertices represent merging of such runs, and an injective function ϕ_w that maps the root of each tree of G_w, representing a set of merging runs, to an element of Q, corresponding to the final state of these right-to-right runs.

In order to represent mergings, we use rooted forests whose vertices are included into $2^Q \setminus \emptyset$. A merging between the right-to-right runs starting from two states q_1 and q_2 is expressed by adding an edge from both $\{q_1\}$ and $\{q_2\}$ towards $\{q_1, q_2\}$. Then, if a third run starting from state $\{q_3\}$ merges with them later on, an edge is added from both $\{q_1, q_2\}$ and $\{q_3\}$ towards $\{q_1, q_2, q_3\}$. Formally, we will use the set $\mathcal{F}_Q$ of rooted forests G satisfying the following properties.

- The vertices of G are non empty subsets of Q.
- The roots of G are disjoint subsets of Q.
- For every vertex s of G, the sons of s are disjoint proper subsets of s.

Note that, as a direct consequence of those properties, there exists no pair of distinct forests of $\mathcal{F}_Q$ that share the same set of vertices. Therefore, in order to define an element of $\mathcal{F}_Q$, it is enough to expose its set of vertices.

In order to ensure that the pair (G_w, ϕ_w) accurately represents the behavior of T on $\vdash w$, we want the two following properties to be satisfied. For every state $q \in Q$ that appears in a vertex of G_w, let $r_q \in 2^Q$ denote the root of the forest G_w that contains q.

$\mathbf{P}_{1.1}$ The vertices of G_w are the sets $s \subseteq Q$ such that the right-to-right runs of T on $\vdash w$ starting in a state belonging to s merge at some point, before merging with any other.

$\mathbf{P}_{1.2}$ For all $(p, q) \in \mathrm{bh}_{rr}(\vdash w)$, we have $\phi_w(r_p) = q$.

We begin by defining $(G_\epsilon, \phi_\epsilon)$. For every $q \in Q$, let $\delta_{\vdash}^{-1}(q) \subseteq Q$ denote the set of states p such that $\delta(p, \vdash) = (\epsilon, q, +1)$. Let G_ϵ be the graph of $\mathcal{F}_Q$ whose set of vertices is composed of the singletons $\{p\}$, and the sets $\delta_{\vdash}^{-1}(q)$ that are not empty. Moreover, let ϕ_ϵ be the function mapping each root $\delta_{\vdash}^{-1}(q)$ of G_ϵ to $q \in Q$. Then both $\mathbf{P}_{1.1}$ and $\mathbf{P}_{1.2}$ are satisfied for $w = \epsilon$.

Now, given $w \in \bar{A}^*$, $a \in \bar{A}$, and a pair (G_w, ϕ_w) satisfying the properties for w, we expose the construction in three steps of the pair (G_{wa}, ϕ_{wa}) satisfying the properties for wa. First, we build a graph G'_{wa} by adding to G_w edges corresponding to the function ϕ_w. Second, we build a graph G''_{wa} by adding to G'_{wa} the local transitions induced by the letter a. Finally, since the intermediate graph G''_{wa} is not an element of $\mathcal{F}_Q$, we shrink it into a new graph $G_{wa} \in \mathcal{F}_Q$. The reduction of this last step is depicted by Figure 5, while the copy $Q^{\mathtt{i}}$ of the second step appear in red on the left picture.

- Let $G'_{wa} = (V'_{wa}, E'_{wa})$ be the graph defined by $V'_{wa} = V_w \cup Q^{\mathtt{i}}$, where $Q^{\mathtt{i}}$ is a copy of the set Q, and $E'_{wa} = E_w \cup \{(\phi_w^{-1}(p), p^{\mathtt{i}}) \in 2^Q \times Q^{\mathtt{i}} | p \in \mathrm{Im}(\phi_w)\}$.

- Let $G''_{wa} = (V''_{wa}, E''_{wa})$ be the graph defined by $V''_{wa} = V'_{wa} \cup Q^{\mathtt{o}}$, where $Q^{\mathtt{o}}$ is a copy of the set Q, and let $E''_{wa} = E'_{wa} \cup \{(p^{\mathtt{i}}, \tau(p)) \in Q^{\mathtt{i}} \times V'_{wa} | p \in Q\}$, where τ is defined as follows. For every $q \in Q$, let s_q denote the smallest vertex of G_w containing q. Then, for every $p \in Q$,

$$\tau(p) = \begin{cases} q^{\mathtt{o}} \in Q^{\mathtt{o}} & \text{if } \delta(p,a) = (q,+1); \\ q^{\mathtt{i}} \in Q^{\mathtt{i}} & \text{if } \delta(p,a) = (q,0); \\ s_q \in V_w & \text{if } \delta(p,a) = (q,-1). \end{cases}$$

- By definition of G''_{wa}, for every vertex $s \in V''_{wa}$ there exists at most one path r_s between s and a vertex $q^{\mathtt{o}} \in Q^{\mathtt{o}}$. Let $\mathsf{out} : V''_{wa} \to Q \cup \{\perp\}$ be the function mapping $s \in V''_{wa}$ to $\perp$ if there exists no such path, and to the state $q \in Q$ corresponding to the target of r_s otherwise. Let $\mathsf{in} : V''_{wa} \to 2^Q$ be the function mapping each vertex s of G''_{wa} to the set $\mathsf{in}(s)$ of states $q \in Q$ such that $\mathsf{out}(q^{\mathtt{i}}) \neq \perp$ and the path $r_{q^{\mathtt{i}}}$ goes through s. Let $G_{wa} \in \mathcal{F}_Q$, be defined by its set of vertices $V_{wa} = \{\mathsf{in}(s) \subset Q | s \in V''_{wa}, \mathsf{in}(s) \neq \emptyset, \text{ and } \mathsf{out}(s) \neq \perp\}$, and let ϕ_{wa} be the function mapping each root $\mathsf{in}(s)$ of V_{wa} to $\mathsf{out}(s) \in Q$.

We prove that the pair (G_{wa}, ϕ_{wa}) satisfies the desired properties.

Lemma 9. *The set of vertices V_{wa} corresponds to a graph $G_{wa} \in \mathcal{F}_Q$. Moreover, the pair (G_{wa}, ϕ_{wa}) satisfies $\mathbf{P}_{1.1}$ and $\mathbf{P}_{1.2}$ for wa.*

Proof. By definition of V_{wa}, the vertices of G_{wa} are non empty subsets of Q. Therefore, in order to show that the set of vertices V_{wa} corresponds to a graph $G_{wa} \in \mathcal{F}_Q$, it is sufficient to show that for every pair of vertices of V_{wa} that are not disjoint, one is included into the other. Let $s_1, s_2 \in V_{wa}$. If s_1 and s_2 are not disjoint, let $s \in s_1 \cap s_2$, let $s'_1 \in V''_{wa}$ denote the state such that $\mathsf{in}(s'_1) = s_1$ and let $s'_2 \in V''_{wa}$ denote the state such that $\mathsf{in}(s'_2) = s_2$. Then there exists a run r_s in G''_{wa} between $q^{\mathtt{i}}$ and an element $q^{\mathtt{o}} \in Q^{\mathtt{o}}$ that goes through s'_1 and s'_2. If s'_1 appears earlier, $s_1 \subseteq s_2$, and if s'_2 appears earlier, $s_2 \subseteq s_1$. This proves the desired result.

Now consider a vertex s of G_{wa}. By construction, this means that all runs starting from the right on $\vdash wa$ in a state q of s end in the same state p. This particularly means that all these runs merge, thus we only have to prove that they merge together before potentially merging with another run. To this end, assume that there exists a state p not present in s, and two states p_1 and p_2 of s such that p merges with p_1 before p_1 merges with p_2. Then either the merging appears while processing w and by induction there is a contradiction, or the mergings appear on a, but by construction of G''_{wa}, this would imply that there is a state containing p and p_1 but not p_2. Hence in both cases we get a contradiction and property $\mathbf{P}_{1.1}$ is satisfied.

Finally, we show that $\mathbf{P}_{1.2}$ is satisfied. By the induction hypothesis, G_w satisfies $\mathbf{P}_{1.2}$, hence for all $(p_1, q_1) \in \mathrm{bh}_{rr}(\vdash w)$, we have $\phi_w(s_{p_1}) = q_1$, where s_{p_1} is the root of G_w containing p_1. Moreover, since $G_w \in \mathcal{F}_Q$, there is a path in G_w between p_1

and s_{p_1}. Therefore, by definition of G'_{wa}, there is a path in G'_{wa} between p_1 and q_1^{in}. Then, by definition of G''_{wa}, for every $(p_2, q_2) \in \mathrm{bh}_{rr}(\vdash wa)$, there is a path r in G''_{wa} between p_2^{in} and q_2^{out}. By definition of G''_{wa}, $\mathsf{in}(p_2^{\mathsf{in}}) \in 2^Q$ is a vertex of G_{wa} that contains p_2. Let s_{p_2} be the root of G_{wa} that contains p_2, and let $s' \in V''_{wa}$ be the state such that $\mathsf{in}(s') = s_{p_2}$. By definition of in, the path r goes through s'. Since q_2^{out} is the target of r, $\phi_{wa}(s_{p_2}) = \mathsf{out}(s') = q_2$, which proves that $\mathbf{P}_{1.2}$ is satisfied. □

Remark 10. *Since the construction of G_{wa} only depends on G_w and a, for every $w' \in \bar{A}^*$, if $G_{w'} = G_w$ then $G_{w'a} = G_{wa}$.*

5.2. *Bounded size complexity*

The next results will allow us to obtain the bound over the size of the set of states presented in the statement of Theorem 8.

Lemma 11. *Every rooted forest $G = (V, E)$ in $\mathcal{F}_Q$ has at most $2|Q| - 1$ vertices.*

Proof. The proof is by induction on the size of Q. If $|Q| = 1$, $|2^Q \setminus \emptyset| = 1$, hence $|V| \leqslant 1 = 2|Q| - 1$. Now suppose that $|Q| > 1$, and that the result is true for every set Q' such that $|Q'| < |Q|$. Let G' be the graph obtained by removing all the roots of G. Then G' is a union of trees $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_m = (V_m, E_m)$. For every $1 \leqslant i \leqslant m$, the forest G_i is an element of $\mathcal{F}_{s_i}$, where $s_i \subset Q$ denotes the root of G_i. Therefore, by using the induction hypothesis, we have

$$\begin{aligned} |V| &= 1 + |V_1| + |V_2| + \ldots + |V_m| \\ &\leqslant 1 + (2|s_1| - 1) + (2|s_2| - 1) + \ldots + (2|s_m| - 1) \\ &= 2(|s_1| + \ldots + |s_m|) + 1 - m \\ &\leqslant 2|Q| - 1. \end{aligned}$$

□

Lemma 12. *Let $n = |Q|$. The size of $\mathcal{F}_Q$ is smaller than or equal to $\frac{(2n)^{2n-2}}{(n-2)!}$.*

Proof. By Cayley's Formula, there exist exactly $(2n)^{2n-2}$ labeled trees on $2n$ vertices. If we only label $n+2$ elements, we gain $n-2$ degrees of liberty and hence there is only $\frac{(2n)^{2n-2}}{(n-2)!}$ such trees. We prove that we can injectively project the elements of $\mathcal{F}_Q$ into trees with $2n$ vertices and $n+2$ labels, proving the bound.

Given an element $G = (V, E)$ of $\mathcal{F}_Q$, we know that $|V| \leqslant 2n - 1$ by Lemma 11. Let G' be the tree on $2n$ vertices obtained by adding to G a vertex $s_\perp$, an edge from each root of G to $s_\perp$, and, if $|V| < 2n - 1$, a linear path composed of $2n - |V| - 1$ new vertices starting from a vertex $s_\top$, and whose end is linked to $s_\perp$. Now for each state q of Q, let s_q be the smallest in terms of inclusion of labels vertice of V that contains q, if it exists. Then each vertex of G' corresponding to a vertex s_q is labeled s_q, and for all q in Q that do not appear in a vertex of V, we label one element of the linear path under $s_\top$ by q. We can assign a different element to all such q by Lemma 11, since q not appearing is equivalent to having V on a smaller set of states.

Thus we proved that G can indeed be embedded into a tree with $2n$ vertices and $n+2$ labels (we added the $s_\perp$ and $s_\top$ labels). We conclude the proof by showing that the projection is injective. Note that all leaves of G are a vertex s_q for some q, and that if all s_q are fixed, then the label of a vertex in G which is not a state s_q is simply the union of the labels of its sons. Then two different forests will be embedded into two different trees, proving injectivity. □

Corollary 13. *Let $n = |Q|$. The size of $\{(G_w, \phi_w) | w \in \bar{A}^*\}$ is smaller than or equal to $(2n)^{2n}$.*

Proof. For every $w \in \bar{A}^*$, since ϕ_w is an injective function mapping the trees of G_w to Q and G_w contains at most n trees, we have $|\{(G_w, \phi_w) | w \in \bar{A}^*\}| \leqslant n! |\mathcal{F}_Q| \leqslant (2n)^{2n}$. □

5.3. *Substitution function*

As a consequence of Lemma 11, for every word $w \in \bar{A}^*$, the graph $G_w = (V_w, E_w)$ admits an injective vertex labeling $\lambda_w : V_w \to \mathcal{X}$, where $\mathcal{X} = (X_1, \ldots, X_{2|Q|-1})$ is a set containing $2|Q|-1$ variables. We now present, for every $w \in A^*$, the construction of a substitution $\sigma_w \in \mathcal{S}_{\mathcal{X},B}$ that will allow us, together with the graph G_w and its vertex labeling λ_w, to describe the output production of the right-to-right runs of T on the input word $\vdash w$. Formally, for every $(p, q) \in \mathrm{bh}_{rr}(\vdash w)$, let $w_{p,q} \in B^*$ denote the production of the corresponding right-to-right run. Moreover, for every vertex s of G_w, let $\bar{\lambda}_w(s)$ denote the concatenation of the λ-labels of the vertices forming the path between s and the root of the corresponding tree in G_w, and for every state $q \in Q$ that appears in at least one vertex of G_w, let $s_q \in 2^Q$ be the vertex of G_w of minimal size such that $q \in s_q$. We want the following property to be satisfied.

P_2 For all $(p, q) \in \mathrm{bh}_{rr}(\vdash w)$, we have $(\sigma_w)(\bar{\lambda}_w(s_p)) = w_{p,q}$.

Let σ_ϵ be the substitution mapping every variable to ϵ. By supposition, for every $p \in Q$, we have $\delta(p, \vdash) = (\epsilon, q, +1)$ for some $q \in Q$. Therefore P_2 is satisfied for ϵ.

Now, given $w \in \bar{A}^*$, $a \in \bar{A}$, and a substitution σ_w satisfying P_2 for w, we expose the construction in three steps of a copyless substitution $\sigma_{w,a}$ such that the composition $\sigma_{wa} = \sigma_w \circ \sigma_{w,a}$ satisfies P_2 for wa. First, we extend the labeling λ_w of V_w to a labeling $\mu''_{wa} : V''_{wa} \to \mathcal{X} \cup B^*$ of G''_{wa}. Second, we reduce it into a labelling $\mu_{wa} : V_{wa} \to (\mathcal{X} \cup B)^*$ of G_{wa}. Then, $\sigma_{w,a}$ is defined as the substitution mapping $\lambda_{wa}(s)$ to $\mu_{wa}(s)$.

- Let $\mu''_{wa} : V''_{wa} \to \mathcal{X} \cup B^*$ be the substitution mapping $s \in V_w$ to $\lambda_w(s)$, $q^{\text{o}} \in Q^{\text{o}}$ to ϵ, and $p^{\text{i}} \in Q^{\text{i}}$ to the word $w' \in B^*$ satisfying $\delta(p, a) = (w', q, m)$.
- For every $t \in V_{wa}$, the set V_t of vertices s of G''_{wa} such that $t = \mathsf{in}(s)$ is not empty, and they form a path $s_1, \ldots, s_m$ such that $s_1 \in Q^{\text{i}} \subseteq V''_{wa}$. Let $\mu_{wa}(t) = \mu''_{wa}(s_1) \ldots \mu''_{wa}(s_m)$.

- For every $X \in \mathcal{X}$, let $\sigma_{w,a}(X) = \epsilon$ if no state $s \in V_{wa}$ is mapped to X by λ_{wa}, and let $\sigma_{w,a}(X) = \mu_{wa}(s)$ for some state $s \in V_w$ satisfying $\lambda_{wa}(s) = X$ otherwise. Since λ_{wa} is injective, the function $\sigma_{w,a}$ is well-defined.

We prove that σ_{wa} satisfies the desired properties.

Lemma 14. *The substitution $\sigma_{w,a}$ is copyless and $\sigma_w \circ \sigma_{w,a}$ satisfies P_2 for wa.*

Proof. We prove that $\sigma_{w,a}$ is copyless by contradiction. Suppose that there exists a variable $X \in \mathcal{X}$ that appears twice in $\sigma_{w,a}(Y)$ for some $Y \in \mathcal{X}$, or that appears in the image of two distinct variables $Y_1, Y_2 \in \mathcal{X}$ by $\sigma_{w,a}$. Then, by definition of $\sigma_{w,a}$, X appears twice in $\mu_{wa}(s)$ for some vertex $s \in V_{wa}$, or X appears in the image of two distinct vertices $s_1, s_2 \in V_{wa}$ by μ_{wa}. Therefore, by definition of μ''_{wa}, there exist two distinct vertices $s'_1, s'_2 \in V''_{wa}$ such that $\lambda_w(s'_1) = \mu''_{wa}(s'_1) = X = \mu''_{wa}(s'_2) = \lambda_w(s'_2)$. This is not possible, since λ_w is injective.

We now prove that P_2 is satisfied. Let $\bar{\mu}_{wa}(s)$ (resp. $\bar{\mu}''_{wa}(s)$) denote the concatenation of the labels of the vertices forming the path starting from a vertex s of G_{wa} (resp. G''_{wa}) and ending at the corresponding root. Since P_2 is satisfied for w by supposition, for every $(p,q) \in \mathrm{bh}_{rr}(wa)$ we have, by definition of G''_{wa}, $\sigma_w(\bar{\mu}_{wa}(\mathsf{in}(p^{\pm}))) = \sigma_w(\bar{\mu}''_{wa}(p^{\pm})) = w_{p,q}$. Note that $\sigma_{wa} = \sigma_w \circ \sigma_{w,a}$, $\mathsf{in}(p^{\pm})$ is equal to the smalles vertex s_p of V_{wa} containing p, and $\sigma_{w,a}(\lambda_{wa}(s_p)) = \mu_{wa}(s_p)$. We obtain $\sigma_{wa}(\bar{\lambda}_{wa}(s_p)) = \sigma_w(\bar{\mu}_{wa}(\mathsf{in}(p^{\pm}))) = w_{p,q}$. This proves that P_2 is satisfied for wa. □

Remark 15. *Since the construction of $\sigma_{w,a}$ only depends on G_w and a, for every $w' \in \bar{A}^*$, if $G_{w'} = G_w$ then $\sigma_{w',a} = \sigma_{w,a}$.*

5.4. *Construction*

We now have all the ingredients to define formally a copyless SST $T' = (A, B, Q', q'_0, Q'_f, \alpha, \mathcal{X}, \beta, F')$ equivalent to T. Recall that we assumed that the 2DFT T read its input starting from the right, and that nothing was produced on the initial marker $\vdash$. Let

- $Q' = \{(G_w, \phi_w) | w \in A^*\}$;
- $q'_0 = (G_\epsilon, \phi_\epsilon)$;
- $Q'_f = \{(G_w, \phi_w) | \phi_w(r_{q_0}) \in F\}$;
- $\alpha : Q' \times A \to Q', ((G_w, \phi_w), a) \mapsto (G_{wa}, \phi_{wa})$;
- $\mathcal{X} = \{X_i | 1 \leqslant i \leqslant 2|Q| - 1\}$;
- $\beta : Q' \times A \to \mathcal{S}_{\mathcal{X},B}, ((G_w, \phi_w), a) \mapsto \sigma_{w,a}$;
- $F' : Q'_f \to (\mathcal{X} \cup B)^*$, $(G_w, \phi_w)) \mapsto \sigma_{w,\dashv}(\bar{\lambda}_{w\dashv}(s_{q_0}))$.

Note that the functions α and β are well-defined as a consequence of the remarks 10 and 15. For every input word $w \in A^*$, as a direct consequence of the definitions of α and β, the state reached by T' on input $\vdash w$ is G_w, and the substitution induced by the corresponding run is σ_w. Therefore, by $\mathsf{P}_{1.1}$ and the definition of P_f, the domains of the function f_T defined by T and the function $f_{T'}$ defined by T' are identical. Moreover, by P_2 and the definition of F', the image of a given word by those two functions are identical, hence the functions are the same.

5.5. *Aperiodicity of the construction*

We now prove that this construction preserves aperiodicity:

Theorem 16. *Let T be an aperiodic* 2DFT. *Then the equivalent* SST *constructed using Theorem 8 is also aperiodic.*

To prove this theorem, we introduce a new equivalence relation $\sim_m$, and we prove that the aperiodicity of the 2DFT T implies the aperiodicity of this relation, which in turn implies the aperiodicity of the SST T'. We say that two words v and w are *merge equivalent* ($v \sim_m w$) if both v and w induce the same merges in the same order, and $bh_{xy}(v) = bh_{xy}(w)$ for every $x, y \in \{r, \ell\}$.

Lemma 17. *Let $v, w \in \bar{A}^*$. If $v \sim_m w$, then for every word $u \in \bar{A}^*$, $G_{uv} = G_{uw}$.*

Proof. This follows from the fact that for every word $u \in \bar{A}^*$, the graph G_{uv} represents the merges of the right-to-right runs of T on uv, and these runs can be decomposed into right-to-right runs on u, and partial runs on v. □

Lemma 18. *Let $w \in \bar{A}^*$ be such that $w^n \sim_T w^{n+1}$. Then $w^{2n} \sim_m w^{2n+1}$.*

Proof. Let w and n be such that $w^n \sim_T w^{n+1}$. Given a run r on w^{2n+1}, if ρ is left-to-left or right-to-right, it never reaches the middle iterations of w in w^{2n+1}. Moreover, if r is left-to-right or right-to-left, it reaches exactly once the middle iteration of w in w^{2n+1}, which it crosses while staying in the same state. This proves that, on input w^{n+1}, runs are only merging in the leftmost and rightmost n iterations of w, and they merge in the same order in w^{2n}. □

The two following lemmas now conclude the proof, since the aperiodicity of both the underlying automaton and the substitution imply the aperiodicity of the whole SST.

Lemma 19. *Let $w \in \bar{A}^*$. If $w^n \sim_T w^{n+1}$, then for every $u \in \bar{A}^*$ $G_{uw^{2n+1}} = G_{uw^{2n}}$.*

Proof. This follows directly from Lemma 18 and Lemma 17. □

Lemma 20. *Let $w \in \bar{A}^*$. If $w^n \sim_T w^{n+1}$, then for every $u \in \bar{A}^*$ $\sigma_{u,w^{2n}} \sim \sigma_{u,w^{2n+1}}$.*

Proof. By construction, reducing the graph G''_{ua} to G_{ua} amounts to deleting the unnecessary information from G_u, i.e. deleting cycles and reducing paths with no new merges to a single vertex. Therefore, each vertex s from G_{ua} can be traced back to a sequence $s_0 \dots s_k$ of vertices of G_u. Should we forget about the production, $\sigma_{w,a}(\lambda_{wa}(s))$ corresponds to $\lambda_w(s_0)\dots\lambda_w(s_k)$. Thanks to Lemma 18, we know that w^n and w^{n+1} are merge equivalent. Then given a state G_u and one of its vertex, it will correspond to the same vertex after reading w^{2n} or w^{2n+1}. Since we have a unique way of mapping vertices to variables, the substitutions $\sigma_{uw^{2n}}$ and $\sigma_{uw^{2n+1}}$ will be equal when production is erased, proving the aperiodicity of the substitution function. □

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

Corollary 21. *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 22. *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 k copies of Y and k copies of Z. We will thus limit, for each variable X, the number of copies of X we maintain. In order to get this information, we will use a look-ahead information on the suffix of the run.

The proof relies on the following fact: suppose that we know at each step what is the substitution induced by the suffix of the run. From this substitution, we know how many copies of each variable X will be involved in the final output. We can thus copy each variable sufficiently many times and use them to produce this substitution in a copyless fashion.

One can observe that there are finitely many substitutions, this information being held in the transition monoid of the SST. Then we can compute, at each step and for each possible substitution, a copyless update. But as a given element of the monoid may have several successors, the update function flows variables from one element to variables of several elements. As these variables are never recombined, we get the 1-boundedness of the construction.

Let $T = (A, B, Q, q_0, Q_f, \delta, \mathcal{X}, \rho, F)$ be an aperiodic k-bounded SST, M_T be its transition monoid, and $\eta_T : A^* \to M_T$ be its transition morphism.

We construct $T' = (A, B, Q', q'_0, Q'_f, \delta', \mathcal{X}', \rho', F')$ where:

- The set of states $Q' = Q \times \mathcal{P}(M_T)$ is the current state plus a set of elements of M_T corresponding to the possible images of the current suffix.
- $q'_0 = (q_0, S_0)$ where $S_0 = \{m \in M_T \mid \delta(q_0, m) \in Q_f\}$ is the set of relevant possible images of input words. Here, we are abusing notations as $\delta(q, m)$ stands for $\delta(q, u)$ where $\eta_T(u) = m$. By definition of the transition monoid, we have that $\eta_T(u) = \eta_T(v)$ implies $\delta(q, u) = \delta(q, v)$, thus this is well defined.
- $Q'_f = \{(q, S) \mid q \in F \text{ and } 1_{M_T} \in S\}$.
- $\delta' : Q' \times A \to Q'$ is defined by $\delta((q, S), a) = (q', S')$ where $q' = \delta(q, a)$ and $S' = \{m \in M_T \mid \eta_T(a)m \in S\}$.
- $\mathcal{X}' = \mathcal{X} \times M_T \times \{1, \dots, k\}$. Variables from $\mathcal{X}'$ will be denoted X_i^m for $X \in \mathcal{X}, i \leqslant k$ and $m \in M_T$.
- The variable update function is defined as follows. First given a state q of T and an element m of M_T, we define $\tilde{\sigma}_{q,m}$ as the projection of the output substitution induced by a run starting on q on a word whose image is m, i.e $\tilde{\sigma}_{q,u} = \gamma(q, u) \circ F(\delta(q, u))$ for $\eta_T(u) = m$. Note that by definition of the transition monoid of an SST, it is well defined.

  Now consider a transition $(q, S) \xrightarrow{a} (q', S')$. For any $s \in S'$ and $0 < i \leqslant |\tilde{\sigma}_{q',s}|_X$, $\rho'((q, S), a, X_i^s)$ is defined similarly to $\rho(q, a, X)$, but all variables are labeled by the element $\eta_T(a)s$ from S and they are all distinct copies to ensure the 1-bounded property. Such a numbering is possible thanks to the fact that s indicates which variables are used as well as how many times. This allows us to copy each variables the right amount of times, using different copies at each occurrence. The k-bounded property then ensures that we will never need more than k variables for a possible output.
- $F' : Q'_f \to (B \cup \mathcal{X}')^*$ is defined as follows. Let $(q, S) \in Q'_f$. The string $F'(q, S)$ is obtained from the string $F(q)$ by substituting each variable X by a variable X_i^n, where $0 < i \leqslant |F(q)|_X$ and $n = 1_{M_T}$. □

Theorem 23. *Let* T *be an aperiodic* k*-bounded* SST*. Then the equivalent* 1*-bounded* SST *constructed using Theorem 22 is also aperiodic.*

Proof. We now have to prove that T' is aperiodic. By construction of T', we have that the runs of T' are of the form $(q, S) \xrightarrow{u} (q', S')$ where $q \xrightarrow{u}_T q'$ and $S' = \{n \in M_T \mid \eta_T(u)n \in S\}$. The update for such a run is then the update of T over the run $q \xrightarrow{u}_T q'$, where variables are labeled by elements from S and S' and numbered accordingly. Then as T is aperiodic, the Q part of the run is also aperiodic by construction. The other part computes sets of runs according to M_T, which is also aperiodic. Then T' will also be aperiodic as the set S' only depends on the image of the word read, and by definition $\eta_T(u^n) = \eta_T(u^{n+1})$ for n large enough. □

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 24. *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 k-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 [7] 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. Inf., 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] R. de Souza. Uniformisation of two-way transducers. In *LATA*, pages 547–558, 2013.
[8] 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.
[9] E. Filiot. Logic-automata connections for transformations. In *ICLA*, volume 8923 of *LNCS*, pages 30–57. Springer, 2015.
[10] E. Filiot, O. Gauwin, and N. Lhote. First-order definability of rational transductions: An algebraic approach. In *LICS*, pages 387–396, 2016.
[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.