SD-Regular Transducer Expressions for Aperiodic Transformations

Luc Dartois, Paul Gastin, Shankara Narayanan

SD-Regular Transducer Expressions for Aperiodic Transformations

Luc Dartois
Univ Paris Est Creteil,
LACL, F-94010 Creteil, France
Email: luc.dartois@lacl.fr

Paul Gastin
Université Paris-Saclay, ENS Paris-Saclay,
CNRS, LMF, 91190, Gif-sur-Yvette, France
Email: paul.gastin@lsv.fr

Shankara Narayanan Krishna
Dept. Computer Science and Engineering,
IIT Bombay, India
Email : krishnas@cse.iitb.ac.in

***Abstract*—FO transductions, aperiodic deterministic two-way transducers, as well as aperiodic streaming string transducers are all equivalent models for first order definable functions. In this paper, we solve the problem of expressions capturing first order definable functions, thereby generalizing the seminal SF=AP (star-free expressions = aperiodic languages) result of Schützenberger. Our result also generalizes a lesser known characterization by Schützenberger of aperiodic languages by SD-regular expressions (SD=AP). We show that every first order definable function over finite words captured by an aperiodic deterministic two-way transducer can be described with an SD-regular transducer expression (SDRTE). An SDRTE is a regular expression where Kleene stars are used in a restricted way: they can appear only on aperiodic languages which are prefix codes of bounded synchronization delay. SDRTEs are constructed from simple functions using the combinators unambiguous sum (deterministic choice), Hadamard product, and unambiguous versions of the Cauchy product and the k-chained Kleene-star, where the star is restricted as mentioned. In order to construct an SDRTE associated with an aperiodic deterministic two-way transducer, (i) we concretize Schützenberger's SD=AP result, by proving that aperiodic languages are captured by SD-regular expressions which are unambiguous and stabilising; (ii) by structural induction on the unambiguous, stabilising SD-regular expressions describing the domain of the transducer, we construct SDRTEs. Finally, we also look at various formalisms equivalent to SDRTEs which use the function composition, allowing to trade the k-chained star for a 1-star.**

I. Introduction

The seminal result of Kleene, which proves the equivalence of regular expressions and regular languages, is among the cornerstones of formal language theory. The Büchi, Elgot, Trakhtenbrot theorem which proved the equivalence of regular languages with MSO definable languages, and the equivalence of regular languages with the class of languages having a finite syntactic monoid, established the synergy between machines, logic and algebra. The fundamental correspondence between machines and logic at the language level has been generalized to transformations by Engelfreit and Hoogeboom [1], where regular transformations are defined by two-way transducers (2DFTs) as well as by the MSO transductions of Courcelle [2]. A generalization of Kleene's theorem to transformations can be found in [3], [4] and [5].

Supported by IRL ReLaX

978-1-6654-4895-6/21/$31.00

In [3], regular transformations were described using additive cost register automata (ACRA) over finite words. ACRAs are a generalization of streaming string transducers (SSTs) [6] which make a single left to right pass over the input and use a finite set of variables over strings from the output alphabet. ACRAs compute partial functions from finite words over a finite input alphabet to a monoid $(\mathbb{D}, +, 0)$. The main contribution of [3] was to provide a set of combinators, analogous to the operators used in regular expressions, which help in forming combinator expressions computing the output of the ACRA over finite words. The result of [3] was generalized to infinite words in [5]. The proof technique in [5] is completely different from [3], and, being algebraic, allows a uniform treatment of the result for transductions over finite and infinite words. Subsequently, an alternative proof for the result of [3] appeared in [4].

The class of star-free languages form a strict subset of regular languages. In 1965, Schützenberger [7] proved his famous result that star-free languages (SF) and languages having an aperiodic syntactic monoid (or aperiodic languages AP) coincide over finite words (SF=AP). This equivalence gives an effective characterization of star-free languages, since one can decide if a syntactic monoid is aperiodic. This was followed by a result [8] of McNaughton and Papert proving the equivalence of star-free languages with counter-free automata as well as first order logic, thereby completing the machine-logic-algebra connection once again. The generalization of this result to transformations has been investigated in [9], [10], proving the equivalence of aperiodic two-way transducers and FO transductions a la Courcelle for finite and infinite words. The counter part of regular expressions for aperiodic languages are star-free expressions, which are obtained by using the complementation operation instead of Kleene-star. The generalization of this result to transformations has been an open problem, as mentioned in [3], [4] and [5]. One of the main challenges in generalizing this result to transformations is the difficulty in finding an analogue for the complementation operation on sets in the setting of transformations.

Our Contributions. The following central problem is our focus: Given an aperiodic 2DFT $\mathcal{A}$, does there exist a class of expressions over basic functions and regular combinators such that, one can effectively compute from $\mathcal{A}$, an expression E in

this class, and conversely, such that $[\![\mathcal{A}]\!](w) = [\![E]\!](w)$ for each $w \in \mathsf{dom}(\mathcal{A})$? We solve this open problem, by providing a characterization by means of expressions for aperiodic two-way transducers. In the following, we describe the main steps leading to the solution of the problem.

Concretizing Schützenberger's characterization. In 1973, Schützenberger [11] presented a characterization of aperiodic languages in terms of SD-expressions which are rational expressions where the star operation is restricted to prefix codes with bounded synchronization delay and no complementation is used. This class of languages is denoted by SD, and this result is known as SD=AP. To circumvent the difficulty of using complementation in star-free expressions, we use this SD=AP characterization of aperiodic languages by SD-expressions. Our *first* contribution is to concretize Schützenberger's result to more specific SD-expressions. We show that aperiodic languages can be captured by *unambiguous*, *stabilising*, SD-expressions. The *unambiguity* of an expression refers to the unique way in which it can be parsed, while *stabilising expressions* is a new notion introduced in this paper. Our concretization, (Theorem III.2) shows that, given a morphism φ from the free monoid Σ^* to a finite aperiodic monoid M, for each $s \in M$, $\varphi^{-1}(s)$ can be expressed by an unambiguous, stabilising SD-expression. The two notions of *unambiguity* and *stabilising* help us to capture the runs of an aperiodic deterministic two-way transducer. These two notions will be described in detail in Section III.

The Combinators. Our *second* contribution is the definition of SD-regular transducer expressions (SDRTE). These are built from basic constant functions using combinators such as unambiguous sum, unambiguous Cauchy product, Hadamard product. In addition, we use k-chained Kleene star $[L, C]^{k\star}$ (and its reverse) when the parsing language L is restricted to be aperiodic and a prefix code with bounded synchronisation delay, and C is an SDRTE. It should be noticed that, contrary to the case of regular transducer expressions (RTE) which define all regular functions, the 2-chained Kleene star $[L, C]^{2\star}$ does not seem sufficient to define all aperiodic functions (see Section IV as well as Figure 2), and k-chained Kleene stars for arbitrary large k seem necessary to capture all aperiodic functions.

The semantics of an SDRTE C is a partial function $[\![C]\!] \colon \Sigma^* \to \Gamma^*$ with domain denoted $\mathsf{dom}(C)$. An SDRTE of the form $L \triangleright v$ where $L \subseteq \Sigma^*$ is an aperiodic language and $v \in \Gamma^*$ is such that $[\![L \triangleright v]\!]$ is a constant function with value v and domain L. The Hadamard product $C_1 \odot C_2$ when applied to $w \in \mathsf{dom}(C_1) \cap \mathsf{dom}(C_2)$ produces $[\![C_1]\!](w) \cdot [\![C_2]\!](w)$. The unambiguous Cauchy product $C_1 \cdot C_2$ when applied on $w \in \Sigma^*$ produces $[\![C_1]\!](u) \cdot [\![C_2]\!](v)$ if w can be unambiguously decomposed as $u \cdot v$, with $u \in \mathsf{dom}(C_1)$ and $v \in \mathsf{dom}(C_2)$. The Kleene star C^* is defined when $L = \mathsf{dom}(C)$ is an aperiodic language which is a prefix code with bounded synchronisation delay. Then $\mathsf{dom}(C^*) = L^*$, and, for the unique decomposition $w = u_1 u_2 \cdots u_n$ with $u_i \in L$, we have $[\![C^*]\!](w) = [\![C]\!](u_1)[\![C]\!](u_2) \cdots [\![C]\!](u_n)$.

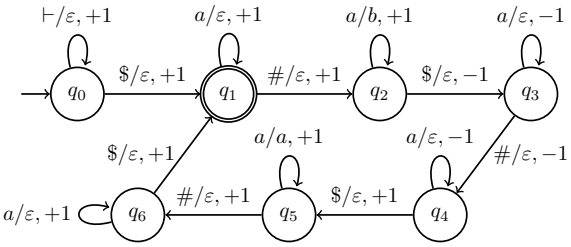


Fig. 1. An aperiodic 2DFT $\mathcal{A}$ with input alphabet $\Sigma = \{a, \#, \$\}$ and output alphabet $\Gamma = \{a, b\}$ computing the partial function $[\![\mathcal{A}]\!](\$a^{m_1}\#a^{m_2}\$a^{m_3}\#a^{m_4}\$ \cdots a^{m_{2k}}\$) = b^{m_2}a^{m_1}b^{m_4}a^{m_3} \cdots b^{m_{2k}}a^{m_{2k-1}}$, for $k \geq 0$.

As an example, consider the SDRTEs $C = C_1 \cdot C_2$, $C' = C'_1 \cdot C'_2$ and $D = C \odot C'$ with $C_1 = (a^*\#) \triangleright \varepsilon$, $C_2 = (a \triangleright b)^* \cdot (\$ \triangleright \varepsilon)$, and $C'_1 = (a \triangleright a)^* \cdot (\# \triangleright \varepsilon)$, $C'_2 = (a^*\$) \triangleright \varepsilon$. Then $\mathsf{dom}(C_1) = a^*\# = \mathsf{dom}(C'_1)$, $\mathsf{dom}(C_2) = a^*\$ = \mathsf{dom}(C'_2)$, and $\mathsf{dom}(C) = a^*\#a^*\$ = \mathsf{dom}(C') = \mathsf{dom}(D)$. Further, $[\![C_1]\!](a^m\#) = \varepsilon$, $[\![C_2]\!](a^n\$) = b^n$, $[\![C'_1]\!](a^m\#) = a^m$, and $[\![C'_2]\!](a^m\$) = \varepsilon$. Also, $[\![D]\!](a^m\#a^n\$) = b^n a^m$. Notice that $\mathsf{dom}(D)^*$ is an aperiodic language (in fact $\mathsf{dom}(D)$ is a prefix code with bounded synchronization delay 1, see Definition 2). Hence, we can define the SDRTE D^* which has domain $\mathsf{dom}(D^*) = (a^*\#a^*\$)^*$, and $[\![D^*]\!](a^2\#a^3\$a^4\#a^5\$) = b^3a^2b^5a^4$. The SDRTE $D' = (\$ \triangleright \varepsilon) \cdot D^*$ corresponds to the aperiodic 2DFT $\mathcal{A}$ in Figure 1: $[\![\mathcal{A}]\!] = [\![D']\!]$.

SDRTE $\leftrightarrow$ Aperiodic 2DFT. Our *third* and main contribution is the effective equivalence between aperiodic two-way transducers and SDRTE*s* over finite words:

Theorem I.1. *(1) Given an SDRTE, we can effectively construct an equivalent aperiodic 2DFT. (2) Given an aperiodic 2DFT, we can effectively construct an equivalent SDRTE.*

The proof of (1) is by structural induction on the SDRTE. All cases except the k-chained Kleene star are reasonably simple, and it is easy to see how to construct the equivalent 2DFT. The case of the k-chained Kleene star is more involved. We write $[L, C]^{k\star}$ as the composition of 3 aperiodic functions f_1, f_2, f_3, where, (i) f_1 takes as input $u_1u_2 \cdots u_n \in L^*$ with $u_i \in L$ and produces as output $\#u_1\#u_2\# \cdots \#u_n\#$, (ii) f_2 takes $\#v_1\#v_2\# \cdots \#v_m\#$ with $v_i \in \Sigma^*$ as input, and produces $\#v_1 \cdots v_k\#v_2 \cdots v_{k+1}\# \cdots \#v_{m-k+1} \cdots v_m\#$ as output, (iii) finally, f_3 takes $\#w_1\#w_2\# \cdots \#w_\ell\#$ as input with $w_i \in \Sigma^*$ and produces as output $[\![C]\!](w_1)[\![C]\!](w_2) \cdots [\![C]\!](w_\ell)$. We produce aperiodic 2DFTs for f_1, f_2, f_3, and compose them, obtaining the required aperiodic 2DFT.

The construction of SDRTE from an aperiodic 2DFT $\mathcal{A}$ over Σ is much more involved, and is based on the transition monoid TrM of the 2DFT $\mathcal{A}$. The translation of $\mathcal{A}$ to SDRTE is guided by unambiguous, stabilising, SD-regular expressions for each element of TrM. These expressions are obtained thanks to Theorem III.2 applied to the canonical morphism $\varphi \colon \Sigma^* \to \mathsf{TrM}$ where the transition monoid TrM of $\mathcal{A}$ is

aperiodic.

Related Work. We provide an aperiodic version of [3]. The combinators we use are essentially the same, with k-chained Kleene star generalising the chained sums of [3]. A natural operation on functions is that of composition. The composition operation can be used in place of the chained-sum operator of [3], and also in place of the unambiguous 2-chained iteration of [5], preserving expressiveness. In yet another recent paper, [12] proposes simple functions like copy, duplicate and reverse along with function composition to capture regular word transductions.

A closely related paper to our work is [13], where first-order and regular list functions were introduced. Using the basic functions reverse, append, co-append, map, block on lists, and combining them with the function combinators of disjoint union, map, pairing and composition, these were shown to be equivalent (after a suitable encoding) to FO transductions a la Courcelle (extendible to MSO transductions by adding to the basic functions, the prefix multiplication operation on groups).

Contrary to [13] where expressions crucially rely on function composition, we focus on concatenation and iteration as first class combinators, in the spirit of Kleene's theorem and of Schützenberger's characterisation AP=SD. We are able to characterise 2DFTs with such SD-regular expressions without using composition. Hence, our result is fully independent and complementary to the work in [13]: both formalisms, SDRTEs and list functions are natural choices for describing first order transductions. Our basic functions and combinators are inspired from the back and forth traversal of a two-way automaton, and the restrictions on the usage of the Kleene star comes from the unambiguous, stabilising nature of the expressions capturing the aperiodic domain of the 2DFT. We also study in Section VI how composition may be used to simplify our SDRTEs (Theorem VI.1). With composition, k-chained Kleene star ($k > 1$) is no more necessary, resulting in an equivalent formalism, namely, SDRTE where we only use 1-star. Yet another equivalent formalism is obtained by restricting SDRTE to simple functions, unambiguous sum, Cauchy product and 1-star, but adding the functions duplicate and reverse along with composition.

Structure of the paper. In Section II, we introduce preliminary notions used throughout the paper. In Section III we give a procedure to construct complement-free expressions for aperiodic languages that suits our approach. This is a generic result on languages, independent of two-way transducers. Section IV presents the combinators and the chain-star operators for our characterization. The main theorem and technical proofs, which is constructing SD-regular transducer expressions from a two-way aperiodic transducer, are in Section V. Finally, the reader can find a running example illustrating the main steps of our approach in [14].

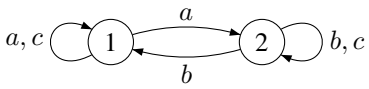
II. Preliminaries

A. Rational languages and monoids

We call a finite set Σ an *alphabet* and its elements *letters*. A finite sequence of letters of Σ is called a *word*, and a set of words is a *language*. The empty word is denoted ε, and we denote by Σ^* the set of all words over the alphabet Σ. More generally, given any language $L \subseteq \Sigma^*$, we write L^* for the *Kleene star* of L, i.e., the set of words which can be written as a (possibly empty) sequence of words of L. Given a word u, we write $|u|$ for the *length* of u, i.e., its number of letters, and we denote by u_i its i^{th} letter.

A *monoid* M is a set equipped with a binary associative law, usually denoted $\cdot$ or omitted when clear from context, and a neutral element 1_M for this law, meaning that for any $s \in M$, $1_M \cdot s = s \cdot 1_M = s$. The set of words Σ^* can be seen as the free monoid generated by Σ using the concatenation of words as binary law. Given a *morphism* $\varphi : \Sigma^* \to M$, i.e., a function between monoids that satisfies $\varphi(\varepsilon) = 1_M$ and $\varphi(xy) = \varphi(x)\varphi(y)$ for any $x, y \in \Sigma^*$, we say that φ recognizes a language $L \subseteq \Sigma^*$ if M is finite and $L = \varphi^{-1}(P)$ for some $P \subseteq M$. A monoid is called *aperiodic* if there exists an integer n such that for any element s of M, $s^n = s^{n+1}$.

Example II.1. *We define the monoids $\widetilde{U}_n$, for $n \geq 0$, as the set of elements $\{1, s_1, \ldots, s_n\}$, with 1 being the neutral element, and for any $1 \leq i, j \leq n$, $s_i \cdot s_j = s_i$. Clearly, $\widetilde{U}_n$ is aperiodic, actually idempotent, as $s_i \cdot s_i = s_i$ for any $1 \leq i \leq n$. For instance, the monoid $\widetilde{U}_2$ is the transition monoid (defined below) of the automaton below with $\varphi(a) = s_1$, $\varphi(b) = s_2$ and $\varphi(c) = 1$.*



Rational languages are languages that can be described by rational expressions, i.e., sets of words constructed from finite sets using the operations of concatenation, union and Kleene star. It is well-known that rational languages are equivalent to *regular* languages, i.e., languages accepted by finite automata, and to languages recognized by finite monoids (and Monadic Second-order logic [15]). *Star-free* rational expressions are built from finite sets using the operations of concatenation, union and complement (instead of Kleene star). They have the same expressive power as finite aperiodic monoids [7] (as well as counter-free automata and first-order logic [8]).

B. Two-way transducers

Definition 1 (Two-way transducer)**.** *A* (deterministic) two-way transducer *(2DFT) is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \gamma, q_0, F)$ where Q is a finite set of* states*, $q_0 \in Q$ is the* initial state*, $F \subseteq Q$ is the set of final states, Σ and Γ are the finite* input *and* output alphabets*, and*

- $\delta : Q \times (\Sigma \uplus \{\vdash, \dashv\}) \to Q \times \{-1, +1\}$ *is the partial* transition function*. Contrary to one-way machines, the transition function also outputs an integer, indicating the move of the reading head. The alphabet is enriched with*

two new symbols $\vdash$ and $\dashv$, which are endmarkers that are added respectively at the beginning and at the end of the input word, such that for all $q \in Q$, we have $\delta(q, \vdash) \in Q \times \{+1\}$ (if defined), $\delta(q, \dashv) \in Q \times \{-1\}$ (if defined) and $\delta(q, \dashv)$ is undefined for $q \in F$.

- $\gamma : Q \times (\Sigma \uplus \{\vdash, \dashv\}) \to \Gamma^*$ *is the partial* production function *with same domain as δ.*

A *configuration* c of $\mathcal{A}$ over an input word $w = w_1 \cdots w_{|w|}$ is simply a pair (p, i) where $p \in Q$ is the current state and $0 \leq i \leq |w| + 1$ is the position of the head on a letter of the input tape containing $\vdash w \dashv$. Two configurations $c = (p, i)$ and $c' = (q, j)$ are *successive* if we have $\delta(p, w_i) = (q, d)$ and $i + d = j$, with $w_0 = \vdash$ and $w_{|w|+1} = \dashv$. In this case, they produce an output $v = \gamma(p, w_i)$. Abusing notations we will sometime write $\gamma(c)$ when the input word w is clear. A run ρ is a sequence of successive configurations $c_0 \cdots c_n$. The run ρ is *initial* if $c_0 = (q_0, 0)$ and is *final* if $c_n = (q, |w| + 1)$ for some $q \in F$. It is *accepting* if it is both initial and final.

The *output* of a run $\rho = c_0 \cdots c_n$ is the concatenation of the output of the configurations, and will be denoted $[\![\rho]\!] = \gamma(c_0) \cdots \gamma(c_{n-1})$. Given a deterministic two-way transducer $\mathcal{A}$ and an input word w, there is at most one accepting run of $\mathcal{A}$ over $\vdash w \dashv$, which we will denote $\rho(w)$. The output of $\mathcal{A}$ over w is then $[\![\mathcal{A}]\!](w) = [\![\rho(w)]\!]$. The *domain* of $\mathcal{A}$ is the set $\mathsf{dom}(\mathcal{A})$ of words w such that there exists an accepting run of $\mathcal{A}$ over w. Finally, the semantics of $\mathcal{A}$ is the partial function $[\![\mathcal{A}]\!] \colon \Sigma^* \to \Gamma^*$ defined on $\mathsf{dom}(\mathcal{A})$ by $w \mapsto [\![\mathcal{A}]\!](w)$.

Let $\rho = (p_0, i_0) \cdots (p_n, i_n)$ be a run over a nonempty word $w \in (\Sigma \cup \{\vdash, \dashv\})^+$ such that $1 \leq i_j \leq |w|$ for all $0 \leq j < n$. It is a *left-right* run if $i_0 = 1$ and $i_n = |w| + 1$. If this is the case, we say that ρ is a $(\rightarrow, p_0, p_n)$-run. Similarly, it is a *left-left* $(\curvearrowright, p_0, p_n)$-run if $i_0 = 1$ and $i_n = 0$. It is a *right-left* $(\leftarrow, p_0, p_n)$-run if $i_0 = |w|$ and $i_n = 0$ and it is a *right-right* $(\subsetneq, p_0, p_n)$-run if $i_0 = |w|$ and $i_n = |w| + 1$. Notice that if $|w| = 1$, then left-right runs and right-right runs coincide, also right-left runs and left-left runs coincide.

Remark 1. *Given our semantics of two-way transducers, a run associates states to each position, whereas the classical semantics of one-way automata keeps the states between two positions. Then, if we consider a word $w = uv$ and a left-left run $(\curvearrowright, p, q)$ on v, we begin on the first position of v in state p, and the state q is reached at the end of the run on the last position of u. This allows for easy sequential composition of partial runs when concatenating non empty words, as the end of a partial run is the start of the next one.*

However, in order to keep our figures as readable as possible, we will represent these states between words. A state q between two words u and v is to be placed on the first position of v if it is the start of a run going to the right, and on the last position of u otherwise. For instance, in Figure 3, state q_1 is on the first position of u_{i+1} and state q_3 is on the last position of u_i.

Transition monoid of a two-way automaton. Let $\mathcal{A}$ be a deterministic two-way automaton (2DFA) with set of states Q. When computing the transition monoid of a two-way automaton, we are interested in the behaviour of the partial runs, i.e., how these partial runs can be concatenated. Thus we abstract a given (d, p, q)-run ρ over a word w to a *step* $(d, p, q) \in \{\rightarrow, \curvearrowright, \subsetneq, \leftarrow\} \times Q^2$ and we say that w realises the step (d, p, q). The transition monoid TrM of $\mathcal{A}$ is a subset of the powerset of steps: $\mathsf{TrM} \subseteq \mathcal{P}(\{\rightarrow, \curvearrowright, \subsetneq, \leftarrow\} \times Q^2)$. The canonical surjective morphism $\varphi \colon (\Sigma \uplus \{\vdash, \dashv\})^* \to \mathsf{TrM} = \varphi((\Sigma \uplus \{\vdash, \dashv\})^*)$ is defined for a word $w \in (\Sigma \uplus \{\vdash, \dashv\})^*$ as the set of steps realised by w, i.e., $\varphi(w) = \{(d, p, q) \mid$ there is a (d, p, q)-run on $w\} \subseteq \{\rightarrow, \curvearrowright, \subsetneq, \leftarrow\} \times Q^2$. As an example, in Figure 1, we have

$$\varphi(a\#) = \{(\rightarrow, q_1, q_2), (\subsetneq, q_1, q_2), (\curvearrowright, q_3, q_3), (\leftarrow, q_3, q_4), (\curvearrowright, q_4, q_4), (\rightarrow, q_5, q_6), (\subsetneq, q_5, q_6)\}.$$

The unit of TrM is $\mathbf{1} = \{(\rightarrow, p, p), (\leftarrow, p, p) \mid p \in Q\}$ and $\varphi(\varepsilon) = \mathbf{1}$.

A 2DFA is *aperiodic* if its transition monoid TrM is aperiodic. Also, a 2DFT is aperiodic if its underlying input 2DFA is aperiodic.

When talking about a given step (d, p, q) belonging to an element of TrM, we will sometimes forget p and q and talk about a d-step, for $d \in \{\curvearrowright, \subsetneq, \rightarrow, \leftarrow\}$ if the states p, q are clear from the context, or are immaterial for the discussion. In this case we also refer to a step (d, p, q) as a d-step having p as the starting state and q as the final state.

III. Complement-free expressions for aperiodic languages

As the aim of the paper is to obtain rational expressions corresponding to transformations computed by aperiodic two-way transducers, we cannot rely on extending the classical (SF=AP) star-free characterization of aperiodic languages, since the complement of a function is not a function. We solve this problem by considering the SD=AP characterization of aperiodic languages, namely prefix codes with bounded synchronisation delay, introduced by Schützenberger [11].

A language L is called a *code* if for any word $u \in L^*$, there is a unique decomposition $u = v_1 \cdots v_n$ such that $v_i \in L$ for $1 \leq i \leq n$. For example, the language $W = \{a, ab, ba, bba\}$ is not a code: the words $abba, aba \in W^*$ have decompositions $a \cdot bba = ab \cdot ba$ and $a \cdot ba = ab \cdot a$ respectively. A *prefix code* is a language L such that for any pair of words u, v, if $u, uv \in L$, then $v = \varepsilon$. W is not a prefix code, while $W_1 = W \setminus \{ab\}$ and $W_2 = W \setminus \{a\}$ are prefix codes. Prefix codes play a particular role since the unique decomposition can be obtained on the fly while reading the word from left to right.

Definition 2. *Let d be a positive integer. A* prefix *code C over an alphabet Σ has a synchronisation delay d (denoted d-SD) if for all $u, v, w \in \Sigma^*$, $uvw \in C^*$ and $v \in C^d$ implies $uv \in C^*$ (hence also $w \in C^*$). An SD prefix code is a prefix code with a bounded synchronisation delay.*

As an example, consider the prefix code $C = \{aa, ba\}$ and the word $ba(aa)^d \in C^*$. We have $ba(aa)^d = uvw$ with $u = b$, $v = (aa)^d \in C^d$ and $w = a$. Since $uv \notin C^*$, the prefix code C

is not of bounded synchronisation delay. Likewise, $C = \{aa\}$ is also not of bounded synchronisation delay. On the other hand, the prefix code $C = \{ba\}$ is 1-SD.

The syntax of regular expressions over the alphabet Σ is given by the grammar

$$E ::= \emptyset \mid \varepsilon \mid a \mid E \cup E \mid E \cdot E \mid E^*$$

where $a \in \Sigma$. We say that an expression is ε-free (resp. $\emptyset$-free) if it does not use ε (resp. $\emptyset$) as subexpressions. The semantics of a regular expression E is a regular language over Σ^* denoted $\mathcal{L}(E)$.

An SD-regular expression is a regular expression where Kleene-stars are restricted to SD prefix codes: If E^* is a sub-expression then $\mathcal{L}(E)$ is a prefix code with bounded synchronization delay. Thus, the regular expression $(ba)^*$ is a SD-regular expression while $(aa)^*$ is not.

The relevance of SD-regular expressions comes from the fact that they are a complement-free characterization of aperiodic languages.

Theorem III.1. *[11] A language L is recognized by an aperiodic monoid if, and only if, there exists an SD-regular expression E such that $L = \mathcal{L}(E)$.*

Theorem III.2 concretizes this result, and extends it to get more specific expressions which are (i) *unambiguous*, a property required for the regular combinators expressing functions over words, and (ii) *stabilising*, which is a new notion introduced below that suits our need for characterizing runs of aperiodic two-way transducers.

A regular expression E is unambiguous, if it satisfies:

1) $\mathcal{L}(E_1) \cap \mathcal{L}(E_2) = \emptyset$ for each subexpression $E_1 \cup E_2$,
2) for each subexpression $E_1 \cdot E_2$, each word $w \in \mathcal{L}(E_1 \cdot E_2)$ has a *unique* factorisation $w = uv$ with $u \in \mathcal{L}(E_1)$ and $v \in \mathcal{L}(E_2)$,
3) for each subexpression E_1^*, the language $\mathcal{L}(E_1)$ is a *code*, i.e., each word $w \in \mathcal{L}(E_1^*)$ has a *unique* factorisation $w = v_1 \cdots v_n$ with $v_i \in \mathcal{L}(E_1)$ for $1 \leq i \leq n$.

Definition 3. *Given an aperiodic monoid M and $X \subseteq M$, we say that X is n-*stabilising *if $xy = x$ for all $x \in X^n$ and $y \in X$. We say that X is stabilising if it is n-stabilising for some $n \geq 1$.*

Remark. Stabilisation generalizes aperiodicity in some sense. For aperiodicity, we require $x^n = x^{n+1}$ for each element $x \in M$ and some $n \in \mathbb{N}$, i.e., all *singleton* subsets of M should be stabilising.

Continuing Example II.1, any subset $X \subseteq \{s_1, \ldots, s_n\} \subseteq \widetilde{U}_n$ is 1-stabilising.

Example III.1. *As another example, consider the aperiodic 2DFT $\mathcal{A}$ in Figure 1, and consider its transition monoid* TrM*. Clearly,* TrM *is an aperiodic monoid. Let φ be the morphism from $(\Sigma \uplus \{\vdash, \dashv\})^*$ to* TrM*. Consider the subset $Z = \{Y, Y^2\}$ of* TrM *where $Y = \varphi(a\#a\$)$:*

$$Y = \{(\rightleftarrows, q_1, q_4), (\rightleftarrows, q_3, q_3), (\rightleftarrows, q_4, q_4), (\rightarrow, q_5, q_1), (\subsetneq, q_0, q_1), (\leftarrow, q_2, q_4), (\subsetneq, q_4, q_5), (\subsetneq, q_6, q_1)\}$$
$$Y^2 = \{(\rightleftarrows, q_1, q_4), (\rightleftarrows, q_3, q_3), (\rightleftarrows, q_4, q_4), (\rightarrow, q_5, q_1), (\subsetneq, q_0, q_1), (\subsetneq, q_2, q_1), (\subsetneq, q_4, q_5), (\subsetneq, q_6, q_1)\}.$$

It can be seen that $Y^3 = Y^2$, hence Z is 2-stabilising.

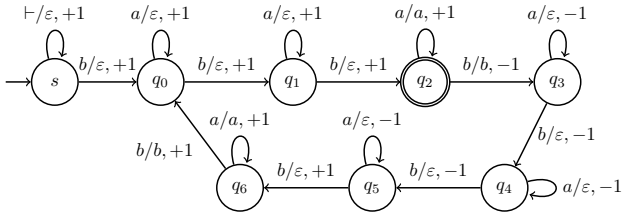


Fig. 2. For $u_i \in a^*b$, an aperiodic 2DFT $\mathcal{A}$ computing the partial function $[\![\mathcal{A}]\!](bu_1u_2\cdots u_na^k) = u_3u_1u_4u_2\cdots u_nu_{n-2}a^k$ if $n \geq 3$, and a^k if $n = 2$. The domain is $b(a^*b)^{\geq 2}a^*$.

Let $\varphi\colon \Sigma^* \to M$ be a morphism. We say that a regular expression E is *φ-stabilising* (or simply stabilising when φ is clear from the context) if for each subexpression F^* of E, the set $\varphi(\mathcal{L}(F))$ is stabilising.

Continuing Example III.1, we can easily see that $\varphi(a)$ is idempotent and we get $\varphi(a^+\#a^+\$) = \{Y\}$. Since $Y^3 = Y^2$, we deduce that $(aa^*\#aa^*\$)^*$ is a stabilising expression. Notice also that, by definition, expressions without a Kleene-star are stabilising vacuously.

A more detailed example for the transducer of Figure 2 is given in [14].

Given a morphism φ from Σ^* to some aperiodic monoid M, our goal is to build, for each language $\varphi^{-1}(s)$ with $s \in M$, an SD-regular expression which is both *unambiguous* and *stabilising*. The proof is by induction on the monoid M via the local divisor technique, similar to Diekert and Kufleitner [16], [17], [18], and to Perrin and Pin [19, Chapter VIII, Section 6.1], with the objective to get stronger forms of SD-regular expressions.

Theorem III.2. *Given a morphism φ from the free monoid Σ^* to a finite aperiodic monoid M, for each $s \in M$ there exists an unambiguous, stabilising, SD-regular expression E_s such that $\mathcal{L}(E_s) = \varphi^{-1}(s)$.*

The proof of this theorem is given in [14], it makes crucial use of *marked substitutions* (see [19]) that we define and study.

IV. Combinator expressions

In this section, we present our combinators to compute first order definable functions from finite words to finite words. The simpler combinators of unambiguous concatenation and sum are similar to those in [3], [5], but we use SD k-chained Kleene-star in lieu of chained-sums.

Simple Functions. For each $v \in \Gamma^*$ we have a constant function f_v defined by $f_v(u) = v$ for all $u \in \Sigma^*$. Abusing

notations, we simply denote the constant function f_v by v. We denote by $\bot\colon \Sigma^* \to \Gamma^*$ the function with empty domain. These atomic functions are the most simple ones.

Unambiguous sums We will use two equivalent ways of defining a function by cases. First, the if-then-else construct is given by $h = L\,?\,f\ :\ g$ where $f, g\colon \Sigma^* \to \Gamma^*$ are functions and $L \subseteq \Sigma^*$ is a language. We have $\mathsf{dom}(h) = (\mathsf{dom}(f) \cap L) \cup (\mathsf{dom}(g) \setminus L)$. Then, for $w \in \mathsf{dom}(h)$ we have

$$h(w) = \begin{cases} f(w) & \text{if } w \in L \\ g(w) & \text{otherwise.} \end{cases}$$

We will often use this case definition with $L = \mathsf{dom}(f)$. To simplify notations we define $f + g = \mathsf{dom}(f)\,?\,f\ :\ g$. Note that $\mathsf{dom}(f+g) = \mathsf{dom}(f) \cup \mathsf{dom}(g)$ but the sum is not commutative and $g + f = \mathsf{dom}(g)\,?\,g : f$. For $w \in \mathsf{dom}(f) \cap \mathsf{dom}(g)$ we have $(f+g)(w) = f(w)$ and $(g+f)(w) = g(w)$. When the domains of f and g are disjoint then $f + g$ and $g + f$ are equivalent functions with domain $\mathsf{dom}(f) \uplus \mathsf{dom}(g)$. In all cases the sum is associative and the sum notation is particularly useful when applied to a sequence $f_1, \ldots, f_n$ of functions:

$$\begin{aligned} \sum_{1 \le i \le n} f_i &= f_1 + \cdots + f_n \\ &= \mathsf{dom}(f_1)\,?\,f_1 : \cdots \mathsf{dom}(f_{n-1})\,?\,f_{n-1} : f_n \end{aligned}$$

If the domains of the functions are pairwise disjoint then this sum is associative and commutative.

Further, we let $L \triangleright f = L\,?\,f\ :\ \bot$ denote the function f restricted to $L \cap \mathsf{dom}(f)$. For a singleton $L = \{w\}$, we write $w \triangleright f$.

The Hadamard product The Hadamard product of two functions $f, g\colon \Sigma^* \to \Gamma^*$ first applies f and then applies g its input. It is denoted by $f \odot g$. Its domain is $\mathsf{dom}(f) \cap \mathsf{dom}(g)$ and $(f \odot g)(u) = f(u)g(u)$ for each input word u in its domain.

The unambiguous Cauchy product Consider two functions $f, g\colon \Sigma^* \to \Gamma^*$. The unambiguous Cauchy product of f and g is the function $f \cdot g$ whose domain is the set of words $w \in \Sigma^*$ which admit a unique factorization $w = uv$ with $u \in \mathsf{dom}(f)$ and $v \in \mathsf{dom}(g)$, and in this case, the computed output is $f(u)g(v)$.

Contrary to the Hadamard product which reads its full input word w twice, first applying f and then applying g, the Cauchy product *splits unamgibuously* its input word w as uv, applies f on u and then g on v.

The k-chained Kleene-star and its reverse Let $L \subseteq \Sigma^*$ be a code, let $k \ge 1$ be a natural number and let $f\colon \Sigma^* \to \Gamma^*$ be a partial function. We define the k-chained Kleene-star $[L, f]^{k\star}\colon \Sigma^* \to \Gamma^*$ and its reverse $[L, f]_r^{k\star}\colon \Sigma^* \to \Gamma^*$ as follows.

The domain of both these functions is contained in L^*, the set of words having a (unique) factorization over the code L. Let $w \in L^*$ and consider its unique factorization $w = u_1 u_2 \cdots u_n$ with $n \ge 0$ and $u_i \in L$ for all $1 \le i \le n$. Then, $w \in \mathsf{dom}([L, f]^{k\star}) = \mathsf{dom}([L, f]_r^{k\star})$ if $u_{i+1} \cdots u_{i+k} \in \mathsf{dom}(f)$ for all $0 \le i \le n - k$ and in this case we set

$$[L, f]^{k\star}(w) = f(u_1 \cdots u_k) \cdot f(u_2 \cdots u_{k+1}) \cdots f(u_{n-k+1} \cdots u_n)$$
$$[L, f]_r^{k\star}(w) = f(u_{n-k+1} \cdots u_n) \cdots f(u_2 \cdots u_{k+1}) \cdot f(u_1 \cdots u_k)\,.$$

Notice that when $n < k$, the right-hand side is an empty product and we get $[L, f]^{k\star}(w) = \varepsilon$ and $[L, f]_r^{k\star}(w) = \varepsilon$. When $k = 1$ and $L = \mathsf{dom}(f)$ is a code then we simply write $f^\star = [\mathsf{dom}(f), f]^{1\star}$ and $f_r^\star = [\mathsf{dom}(f), f]_r^{1\star}$. We have $\mathsf{dom}(f^\star) = \mathsf{dom}(f_r^\star) = L^*$.

The k-chained Kleene star was also defined in [3], [5]; however as we will see below, we use it in a restricted way for aperiodic functions.

SD-regular transducer expressions (SDRTE) SD-regular transducer expressions (SDRTEs) are obtained from classical regular transducer expressions (RTEs) [3], [5] by restricting the k-chained Kleene-star $[L, f]^{k\star}$ and its reverse $[L, f]_r^{k\star}$ to aperiodic languages L that are prefix codes of bounded synchronisation delay. The if-then-else choice $L\,?\,f\ :\ g$ is also restricted to aperiodic languages L. Hence, the syntax of SDRTEs is given by the grammar:

$$C ::= \bot \mid v \mid L\,?\,C : C \mid C \odot C \mid C \cdot C \mid [L, C]^{k\star} \mid [L, C]_r^{k\star}$$

where $v \in \Gamma^*$, and $L \subseteq \Sigma^*$ ranges over aperiodic languages (or equivalently SD-regular expressions), which are also prefix codes with bounded synchronisation delay for $[L, C]^{k\star}$ and $[L, C]_r^{k\star}$.

The semantics of SDRTEs is defined inductively. $[\![\bot]\!]$ is the function which is nowhere defined, $[\![v]\!]$ is the constant function such as $[\![v]\!](u) = v$ for all $u \in \Sigma^*$, and the semantics of the other combinators has been defined in the above sections.

As discussed in Section IV, we will use binary sums $C + C' = \mathsf{dom}(C)\,?\,C : C'$ and generalised sums $\sum_i C_i$. Also, we use the abbreviation $L \triangleright C = L\,?\,C : \bot$.

We now give a serie of technical properties of SDRTEs that will be used in the following sections. The proofs are given in [14].

Lemma 1. *If C is an SDRTE, then $\mathsf{dom}(C)$ is an aperiodic language.*

Proposition 1. *Given an SDRTE C and a letter $a \in \Sigma$,*

1) *we can construct an SDRTE $a^{-1}C$ such that $\mathsf{dom}(a^{-1}C) = a^{-1}\mathsf{dom}(C)$ and $[\![a^{-1}C]\!](w) = [\![C]\!](aw)$ for all $w \in a^{-1}\mathsf{dom}(C)$,*
2) *we can construct an SDRTE Ca^{-1} such that $\mathsf{dom}(Ca^{-1}) = \mathsf{dom}(C)a^{-1}$ and $[\![Ca^{-1}]\!](w) = [\![C]\!](wa)$ for all $w \in \mathsf{dom}(C)a^{-1}$.*

Lemma 2. *Given an SDRTE C over an alphabet Σ and a sub-alphabet $\Sigma' \subseteq \Sigma$, we can construct an SDRTE C'* over alphabet Σ' *such that $\mathsf{dom}(C') \subseteq \Sigma'^*$ and for any word w in Σ'^*, $[\![C]\!](w) = [\![C']\!](w)$.*

Can the 2-chained Kleene star suffice for all aperiodic functions? It is known [5] that the 2-chained Kleene star can

simulate the k-chained Kleene-star for regular functions. However, we believe that, contrary to the case of regular functions, the k-chained Kleene-star operator cannot be simulated by the 2-chained Kleene-star while preserving the aperiodicity of the expression. The key idea is that, in order to simulate a k-chained Kleene-star on a SD prefix code L using a 2-chained Kleene-star, one needs to use $L^{\lceil k/2 \rceil}$ as a parser. However, for any given prefix code L, the language L^n for $n > 1$, while still a prefix code, is not of bounded synchronisation delay (for the same reason that $\{aa\}$ is not, i.e., for $v = (aa)^d$ that we consider, ava belongs to $(aa)^*$ but av does not). Intuitively, parsing L^n reduces to *count*ing factors of L *modulo* n, which is a classical example of non-aperiodicity.

As an example, consider the prefix code $L = (a + b)^*c$ which has synchronisation delay 1. Define a function f with domain L^3 by $f(u_1u_2u_3) = u_3u_1$ when $u_1, u_2, u_3 \in L$, which can be written using combinators as $\big((L^2 \triangleright \varepsilon) \cdot (L \triangleright id)\big) \odot \big((L \triangleright id) \cdot (L^2 \triangleright \varepsilon)\big)$. The identity function id can itself be written as $(a \triangleright a + b \triangleright b + c \triangleright c)^\star$ (see also Figure 2, which is a simplification of the same function, but neverthless has the same inexpressiveness with 2 chained star). Then we believe that the function $[L, f]^{3\star}$, which associates to a word $u_1 \cdots u_n \in L^*$ the word $u_3u_1u_4u_2 \cdots u_nu_{n-2}$ is not definable using only 2-chained Kleene-stars. While not a proof, the intuition behind this is that, in order to construct $u_{i+1}u_{i-1}$, we need to highlight words from L^3. In order to do this with a 2-chained Kleene-star, it seems necessary to apply a chained star with parser L^2, which is a prefix code but not of bounded synchronisation delay. A similar argument would hold for any $[L, f]^{k\star}$, $k \geq 3$ with a function $f(u_1u_2 \cdots u_k) = u_ku_1$.

V. The Equivalence of SDRTE and Aperiodic 2DFT

In this section, we prove the main result of the paper, namely the equivalence between SDRTE and aperiodic 2DFT stated in Theorem I.1. The first direction, given an SDRTE C, constructing an equivalent aperiodic 2DFT $\mathcal{A}$ is given by Theorem V.1, while Theorem V.2 handles the converse.

A. Aperiodic 2DFTs for SD-regular transducer expressions

Theorem V.1. *Given an SDRTE C, we can construct an equivalent aperiodic 2DFT $\mathcal{A}$ with $[\![C]\!] = [\![\mathcal{A}]\!]$.*

Proof. We construct $\mathcal{A}$ by induction on the structure of the SDRTE C. In the suitable cases, we will suppose thanks to induction that we have aperiodic transducers $\mathcal{A}_i$ for expressions C_i, $i \leq 2$. We also have a deterministic and complete aperiodic automaton A_L for any aperiodic language L.

- $C = \bot$. Then $\mathcal{A}$ is a single state transducer with no final state so that its domain is empty.
- $C = v$. Then $\mathcal{A}$ is a single state transducer which produces v and accepts any input word. Clearly, $\mathcal{A}$ is aperiodic.
- $C = L\,?\,C_1 : C_2$. The transducer $\mathcal{A}$ first reads its input, simulating A_L. Upon reaching the end of the input word, it goes back to the beginning of the word, and either executes $\mathcal{A}_1$ if the word was accepted by A_L, or executes $\mathcal{A}_2$ otherwise. Since every machine was aperiodic, so is $\mathcal{A}$.
- $C = C_1 \odot C_2$. The transducer $\mathcal{A}$ does a first pass executing $\mathcal{A}_1$, then resets to the beginning of the word and simulates $\mathcal{A}_2$. Since both transducers are aperiodic, so is $\mathcal{A}$.
- $C = C_1 \cdot C_2$. We express $\mathcal{A}$ as the composition of three functions f_1, f_2, f_3, each aperiodic. Since aperiodic functions are closed under composition, we get the result. The first function f_1 associates to each word $w \in \Sigma^*$ the word $u_1\#u_2\#\cdots\#u_n$, such that $w = u_1u_2\cdots u_n$ and for any prefix u of w, u belongs to the domain of C_1 if, and only if, $u = u_1 \cdots u_i$ for some $1 \leq i < n$. Notice that $u_1 = \varepsilon$ iff $\varepsilon \in \mathsf{dom}(C_1)$ and $u_n = \varepsilon$ iff $w \in \mathsf{dom}(C_1)$. The other u_i's must be nonempty. The second function f_2 takes as input a word in $(\Sigma \cup \{\#\})^*$, reads it from right to left, and suppresses all $\#$ symbols except for the ones whose corresponding suffix belongs to the domain of C_2. Then, $f_2(f_1(w))$ contains exactly one $\#$ symbol if and only if w has a unique factorisation $w = uv$ with $u \in \mathsf{dom}(C_1)$ and $v \in \mathsf{dom}(C_2)$. In this case, $f_2(f_1(w)) = u\#v$.

  Finally, the function f_3 has domain $\Sigma^*\#\Sigma^*$ and first executes $\mathcal{A}_1$ on the prefix of its input upto the $\#$ symbol, treating it as the right endmarker $\dashv$, and then executes $\mathcal{A}_2$ on the second part, treating $\#$ as the left endmarker $\vdash$.

  The functions f_1 and f_2 can be realised by aperiodic transducers as they only simulate automata for the aperiodic domains of C_1 and the reverse of C_2 respectively, and the function f_3 executes $\mathcal{A}_1$ and $\mathcal{A}_2$ one after the other, and hence is also aperiodic.
- $C = [L, C_1]^{k\star}$ or $C = [L, C_1]_r^{k\star}$. Here $L \subseteq \Sigma^*$ is an aperiodic language which is also a prefix code with bounded synchronisation delay, and $k \geq 1$ is a natural number. Let $f = [\![C_1]\!] \colon \Sigma^* \to \Gamma^*$ be the aperiodic function defined by C_1. We write $[L, f]^{k\star} = L^{<k}\,?\,(\Sigma^* \triangleright \varepsilon) : (f_3 \circ f_2 \circ f_1)$, where ε is the output produced when the input has less than k L factors; otherwise the output is produced by the composition of 3 aperiodic functions. As aperiodic functions are closed under composition, this gives the result. The first one, $f_1 \colon \Sigma^* \to (\Sigma \cup \{\#\})^*$ splits an input word $w \in L^*$ according to the unique factorization $w = u_1u_2\cdots u_n$ with $n \geq 0$ and $u_i \in L$ for all $1 \leq i \leq n$ and inserts $\#$ symbols: $f_1(w) = \#u_1\#u_2\#\cdots\#u_n\#$. The domain of f_1 is L^*.

  The second function f_2 constructs the sequence of k factors. Its domain is $\#(\Sigma^*\#)^{\geq k}$ and it is defined as follows, with $u_i \in \Sigma^*$: $f_2(\#u_1\#u_2\#\cdots\#u_n\#) =$

$$\#u_1u_2\cdots u_k\#u_2\cdots u_{k+1}\#\cdots\#u_{n-k+1}\cdots u_n\#\,.$$

  Finally, the third function simply applies f and erases the $\#$ symbols:

$$f_3(\#v_1\#v_2\#\cdots\#v_m\#) = f(v_1)f(v_2)\cdots f(v_m)\,.$$

  In particular, $f_3(\#) = \varepsilon$. We have $\mathsf{dom}(f_3) = \#(\mathsf{dom}(f)\#)^*$.

  For the reverse iteration, we simply change the last function and use instead

$$f_4(\#v_1\#v_2\#\cdots\#v_m\#) = f(v_m)\cdots f(v_2)f(v_1)\,.$$

It is easy to see that the functions f_i for $i \leq 4$ can be realised by 2DFTs. We prove in [14] Lemma 18 that these functions are realised by aperiodic 2DFTs. □

B. Stabilising runs in aperiodic two-way automata

In this section, we show that runs of an aperiodic 2DFT have a "stabilising" property. This property crucially distinguishes aperiodic 2DFTs from non aperiodic ones, and we use this in our proof to obtain SDRTEs from aperiodic 2DFTs. In the remainder of this section, we fix an aperiodic 2DFT $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \gamma, q_0, F)$. Let $\varphi\colon (\Sigma \uplus \{\vdash, \dashv\})^* \to \mathsf{TrM}$ be the canonical surjective morphism to the transition monoid of $\mathcal{A}$.

Consider a *code* $L \subseteq \Sigma^*$ such that $X = \varphi(L)$ is k-stabilizing for some $k > 0$. We will see that a run of $\mathcal{A}$ over a word $w \in L^*$ has some nice properties. Intuitively, if it moves forward through k factors from L then it never moves backward through more than k factors. An illustration of these nice properties is given in [14].

More precisely, let $w = u_1 u_2 \cdots u_n$ be the unique factorisation of $w \in L^*$ with $u_i \in L$ for $1 \leq i \leq n$. We assume that $n \geq k$. We start with the easiest fact.

Lemma 3. *If $(\circlearrowright, p, q) \in \varphi(w)$ then the run of $\mathcal{A}$ over w starting on the left in state p only visits the first k factors $u_1 \cdots u_k$ of w.*

Proof. Since X is k-stabilising, we have $\varphi(w) = \varphi(u_1 \cdots u_k)$. Hence, $(\circlearrowright, p, q) \in \varphi(u_1 \cdots u_k)$ and the result follows since $\mathcal{A}$ is deterministic. □

Notice that the right-right ($\circlearrowleft$) runs of $\mathcal{A}$ over w need not visit the last k factors only (see Lemma 6 below). This is due to the fact that *stabilising* is not a symmetric notion.

Next, we consider the left-right runs of $\mathcal{A}$ over w.

Lemma 4. *Assume that $(\rightarrow, p, q) \in \varphi(w)$. Then the run ρ of $\mathcal{A}$ over w starting on the left in state p has the following property, that we call k-forward-progressing: for each $1 \leq i < n - k$, after reaching the suffix $u_{i+k+1} \cdots u_n$ of w, the run ρ will never visit again the prefix $u_1 \cdots u_i$. See Figure 3 for a non-example and Figure 7 for an example.*

Proof. Towards a contradiction, assume that for some $1 \leq i < n-k$, the run ρ visits $u_1 \cdots u_i$ after visiting $u_{i+k+1} \cdots u_n$ (See Figure 3). Then, there exists a subrun ρ' of ρ making some $(\circlearrowright, q_1, q_3)$-step on $u_{i+1} \cdots u_n$ and visiting u_{i+k+1} (on Figure 3 we have $\rho' = \rho_2\rho_3$). Hence $(\circlearrowright, q_1, q_3) \in \varphi(u_{i+1} \cdots u_n)$ and by Lemma 3 we deduce that ρ' visits $u_{i+1} \cdots u_{i+k}$ only, a contradiction. □

Lemma 5. *Assume that $(\leftarrow, p, q) \in \varphi(w)$. Then the run ρ of $\mathcal{A}$ over w starting on the right in state p has the following property, that we call k-backward-progressing: for each $1 \leq i < n - k$, after reaching the prefix $u_1 \cdots u_i$ of w, the run ρ will never visit again the suffix $u_{i+k+1} \cdots u_n$.*

Proof. This Lemma is a consequence of Lemma 3. Indeed, consider any part of ρ that visits u_{i+1} again (in some state q_1) after visiting u_i, for some $1 \leq i < n - k$. As ρ is a $\leftarrow$ run, it will later cross from u_{i+1} to u_i (reaching some state q_3). Then $(\circlearrowright, q_1, q_3)$ is a run on $u_{i+1} \cdots u_n$. By Lemma 3, it does not visit $u_{i+k+1} \cdots u_n$, which concludes the proof (See Figure 4 for a non-example). □

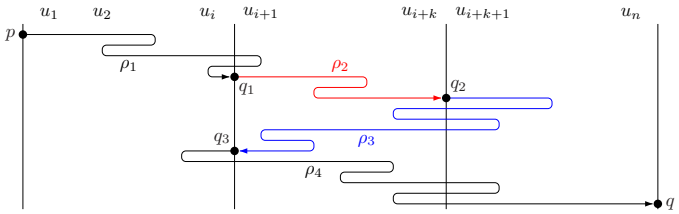


Fig. 3. A left-right run which is not k-forward-progressing

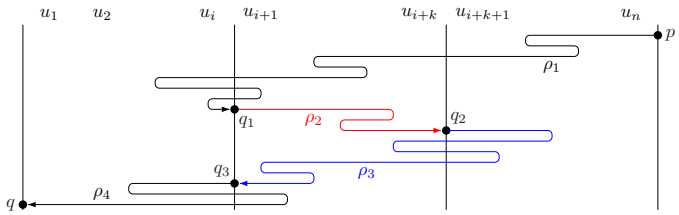


Fig. 4. A right-left run which is not k-backward-progressing

Lemma 6. *Assume that $(\circlearrowleft, p, q) \in \varphi(w)$ and let ρ be the run of $\mathcal{A}$ over w starting on the right in state p. Then, either ρ visits only the last k factors $u_{n-k+1} \cdots u_n$, or for some $1 \leq i \leq n - k$ the run ρ is the concatenation $\rho_1\rho_2\rho_3$ of a k-backward-progressing run ρ_1 over $u_{i+1} \cdots u_n$ followed by a run ρ_2 staying inside some $u_i \cdots u_{i+k}$, followed by some k-forward-progressing run ρ_3 over $u_{i+1} \cdots u_n$. See Figure 5.*

Proof. Assume that ρ visits $u_1 \cdots u_{n-k}$ and let u_i ($1 \leq i \leq n-k$) be the left-most factor visited by ρ. We split ρ in $\rho_1\rho_2\rho_3$ (see Figure 5) where

- ρ_1 is the prefix of ρ, starting on the right of w in state p and going until the first time ρ crosses from u_{i+1} to u_i. Hence, ρ_1 is a run over $u_{i+1} \cdots u_n$ starting on the right in state p and exiting on the left in some state q_1. We have $(\leftarrow, p, q_1) \in \varphi(u_{i+1} \cdots u_n)$. By Lemma 5 we deduce that ρ_1 is k-backward-progressing.
- Then, ρ_2 goes until the last crossing from u_i to u_{i+1}.
- Finally, ρ_3 is the remaining suffix of ρ. Hence, ρ_3 is a run over $u_{i+1} \cdots u_n$ starting on the left in some state q_2 and exiting on the right in state q. We have $(\rightarrow, q_2, q) \in \varphi(u_{i+1} \cdots u_n)$. By Lemma 4 we deduce that ρ_3 is k-forward-progressing.

It remains to show that ρ_2 stays inside $u_i \cdots u_{i+k}$. Since u_i is the left-most factor visited by ρ, we already know that ρ_2 does not visit $u_1 \cdots u_{i-1}$. Similarly to Lemma 5, any maximal subrun ρ_2' of ρ_2 that does not visit u_i is a $\circlearrowright$ run on $u_{i+1} \cdots u_n$ since ρ_2 starts and ends at the frontier between u_i and u_{i+1}. By Lemma 3, the subrun ρ_2' does not visit $u_{i+k+1} \cdots u_n$ and thus ρ_2 stays inside $u_i \cdots u_{i+k}$. □

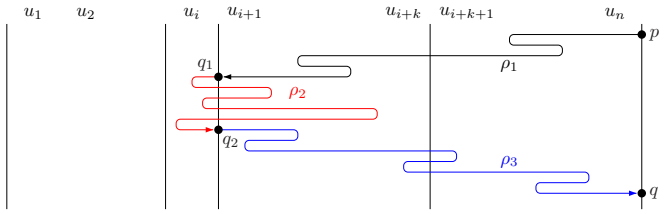


Fig. 5. A right-right run $\rho_1\rho_2\rho_3$ where ρ_1 is k-backward-progressing, ρ_2 is local to $u_i \cdots u_{i+k}$ and ρ_3 is k-forward-progressing.

C. SD-regular transducer expressions for aperiodic 2DFTs

In this section, we show how to construct SDRTEs which are equivalent to aperiodic 2DFTs. Recall that $\varphi\colon (\Sigma \uplus \{\vdash, \dashv\})^* \to \mathsf{TrM}$ is the canonical surjective morphism to the transition monoid of the 2DFT $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \gamma, q_0, F)$. Given a regular expression E and a monoid element $s \in \mathsf{TrM}$, we let $\mathcal{L}(E, s) = \mathcal{L}(E) \cap \varphi^{-1}(s)$. The main construction of this section is given by Theorem V.2. It relies on the transition monoid of 2DFT that is given in Section II.

Theorem V.2. *Let E be an unambiguous, stabilising, SD-regular expression over $\Sigma \uplus \{\vdash, \dashv\}$ and let $s \in \mathsf{TrM}$. For each step $x \in \{\rightarrow, \curvearrowright, \subsetneq, \leftarrow\} \times Q^2$, we can construct an SDRTE $\mathsf{C}_{E,s}(x)$ such that:*

1) *$\mathsf{C}_{E,s}(x) = \bot$ when $x \notin s$, and otherwise*
2) *$\mathsf{dom}([\![\mathsf{C}_{E,s}(x)]\!]) = \mathcal{L}(E, s)$ and for all words $w \in \mathcal{L}(E, s)$, $[\![\mathsf{C}_{E,s}(x)]\!](w)$ is the output produced by $\mathcal{A}$ running over w according to step x.*
   When $w = \varepsilon$ and $s = \mathbf{1} = \varphi(\varepsilon)$ with $x \in \mathbf{1}$, this means $[\![\mathsf{C}_{E,s}(x)]\!](\varepsilon) = \varepsilon$.

Proof. The construction is by structural induction on E.

Atomic expressions We first define $\mathsf{C}_{E,s}(x)$ when E is an atomic expression, i.e., $\emptyset$, ε or a for $a \in \Sigma$.

• $E = \emptyset$: we simply set $\mathsf{C}_{\emptyset,s}(x) = \bot$, which is the nowhere defined function.

• $E = \varepsilon$: when $s = \mathbf{1}$ and $x \in s$ then we set $\mathsf{C}_{\varepsilon,s}(x) = \varepsilon \triangleright \varepsilon$ and otherwise we set $\mathsf{C}_{\varepsilon,s}(x) = \bot$.

• $E = a \in \Sigma \uplus \{\vdash, \dashv\}$: again, we set $\mathsf{C}_{a,s}(x) = \bot$ if $s \neq \varphi(a)$ or $x \notin s$. Otherwise, there are two cases. Either $x \in \{(\rightarrow, p, q), (\subsetneq, p, q)\}$ for some states p, q such that $\delta(p, a) = (q, +1)$, or $x \in \{(\leftarrow, p, q), (\curvearrowright, p, q)\}$ for some states p, q with $\delta(p, a) = (q, -1)$. In both cases the output produced is $\gamma(p, a)$ and we set $\mathsf{C}_{a,s}(x) = a \triangleright \gamma(p, a)$.

Disjoint union Consider $E \cup F$ with $\mathcal{L}(E)$ and $\mathcal{L}(F)$ disjoint, then we simply set $\mathsf{C}_{E\cup F,s}(x) = \mathsf{C}_{E,s}(x) + \mathsf{C}_{F,s}(x)$.

Unambiguous concatenation $E \cdot F$

Here, we suppose that we have SDRTEs for $\mathsf{C}_{E,s}(x)$ and $\mathsf{C}_{F,s}(x)$ for all s in TrM and all steps $x \in \{\rightarrow, \curvearrowright, \subsetneq, \leftarrow\} \times Q^2$. We show how to construct SDRTEs for $\mathsf{C}_{E\cdot F,r}(x)$ for all elements $r \in \mathsf{TrM}$, assuming that the concatenation $\mathcal{L}(E) \cdot \mathcal{L}(F)$ is unambiguous.

A word $w \in \mathcal{L}(E \cdot F)$ has a unique factorization $w = uv$ with $u \in \mathcal{L}(E)$ and $v \in \mathcal{L}(F)$. Let $s = \varphi(u)$ and $t = \varphi(v)$. A run ρ over w is obtained by stitching together runs over u and runs over v as shown in Figure 6. In the top figure, the run over w follows step $x = (\rightarrow, p, q)$ starting on the left in state p and exiting on the right in state q. The run ρ splits as $\rho_0\rho_1\rho_2\rho_3\rho_4\rho_5$ as shown in the figure. The output of the initial part ρ_0 is computed by $\mathsf{C}_{E,s}((\rightarrow, p, p_1))$ over u and the output of the final part ρ_5 is computed by $\mathsf{C}_{F,t}((\rightarrow, p_5, q))$ over v. We focus now on the internal part $\rho_1\rho_2\rho_3\rho_4$ which consists of an alternate sequence of left-left runs over v and right-right runs over u. The corresponding sequence of steps $x_1 = (\curvearrowright, p_1, p_2) \in t$, $x_2 = (\subsetneq, p_2, p_3) \in s$, $x_3 = (\curvearrowright, p_3, p_4) \in t$ and $x_4 = (\subsetneq, p_4, p_5) \in s$ depends only on $s = \varphi(u)$ and $t = \varphi(v)$.

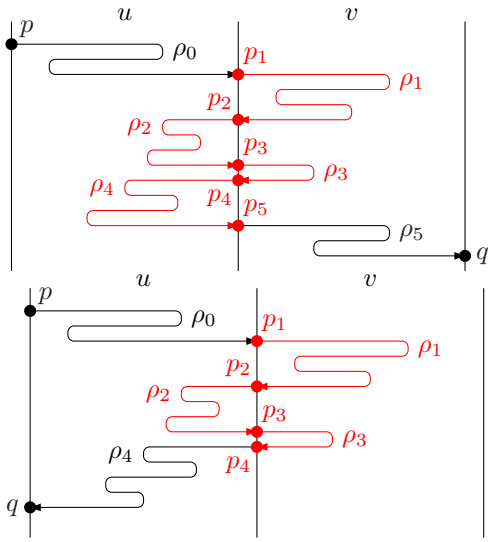


Fig. 6. Decomposition of a $(\rightarrow, p, q)$-run and a $(\curvearrowright, p, q)$-run over the product $w = uv$.

These internal zigzag runs will be frequently used when dealing with concatenation or Kleene star. They alternate left-left ($\curvearrowright$) steps on the right word v and right-right ($\subsetneq$) steps on the left word u. They are also uniquely determined by the starting configuration thanks to the determinism of $\mathcal{A}$. They may start with a $\curvearrowright$-step or a $\subsetneq$-step. The sequence of steps in a *maximal* zigzag run is entirely determined by the monoid elements $s = \varphi(u)$, $t = \varphi(v)$, the starting step $d \in \{\curvearrowright, \subsetneq\}$ and the starting state p' of step d. The final step of this *maximal* sequence is some $d' \in \{\curvearrowright, \subsetneq\}$ and reaches some state q'. We write $Z_{s,t}(p', d) = (d', q')$. For instance, for Figure 6 (top) we get $Z_{s,t}(p_1, \curvearrowright) = (\subsetneq, p_5)$ whereas for Figure 6 (bottom) we get $Z_{s,t}(p_1, \curvearrowright) = (\curvearrowright, p_4)$. By convention, if the sequence of zigzag steps is empty then we define $Z_{s,t}(p, \curvearrowright) = (\subsetneq, p)$ and $Z_{s,t}(p, \subsetneq) = (\curvearrowright, p)$. The construction is illustrated in [14].

Lemma 7. *We use the above notation. We can construct SDRTEs $\mathsf{ZC}^{F,t}_{E,s}(p, d)$ for $p \in Q$ and $d \in \{\curvearrowright, \subsetneq\}$ such that $\mathsf{dom}([\![\mathsf{ZC}^{F,t}_{E,s}(p, d)]\!]) = \mathcal{L}(E, s)\mathcal{L}(F, t)$ and for all $u \in \mathcal{L}(E, s)$ and $v \in \mathcal{L}(F, t)$ the value $[\![\mathsf{ZC}^{F,t}_{E,s}(p, d)]\!](uv)$ is the output produced by the* internal *zigzag run of $\mathcal{A}$ over (u, v) following the maximal sequence of steps starting in state p with a d-step.*

Proof. We first consider the case $d = \rightsquigarrow$ and $Z_{s,t}(p, \rightsquigarrow) = (\subsetneq, q)$ for some $q \in Q$ which is illustrated on the left of Figure 6. Since $\mathcal{A}$ is deterministic, there is a unique maximal sequence of steps (with $n \geq 0$, $p_1 = p$ and $p_{2n+1} = q$): $x_1 = (\rightsquigarrow, p_1, p_2) \in t$, $x_2 = (\subsetneq, p_2, p_3) \in s$, ..., $x_{2n-1} = (\rightsquigarrow, p_{2n-1}, p_{2n}) \in t$, $x_{2n} = (\subsetneq, p_{2n}, p_{2n+1}) \in s$. The zigzag run ρ following this sequence of steps over uv splits as $\rho_1\rho_2\cdots\rho_{2n}$ where ρ_{2i} is the unique run on u following step x_{2i} and ρ_{2i+1} is the unique run on v following step x_{2i+1}. The output of these runs are given by $[\![\mathsf{C}_{E,s}(x_{2i})]\!](u)$ and $[\![\mathsf{C}_{F,t}(x_{2i+1})]\!](v)$. When $n = 0$ the zigzag run ρ is empty and we simply set $\mathsf{ZC}^{F,t}_{E,s}(p, \rightsquigarrow) = (\mathcal{L}(E,s)\mathcal{L}(F,t)) \triangleright \varepsilon$. Assume now that $n > 0$. The required SDRTE computing the output of ρ can be defined as $\mathsf{ZC}^{F,t}_{E,s}(p, \rightsquigarrow) =$

$$\big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}(x_1)\big) \odot \big(\mathsf{C}_{E,s}(x_2) \cdot \mathsf{C}_{F,t}(x_3)\big) \odot \cdots \odot \big(\mathsf{C}_{E,s}(x_{2n-2}) \cdot \mathsf{C}_{F,t}(x_{2n-1})\big) \odot \big(\mathsf{C}_{E,s}(x_{2n}) \cdot (\mathcal{L}(F,t) \triangleright \varepsilon)\big).$$

Notice that each Cauchy product in this expression is unambiguous since the product $\mathcal{L}(E) \cdot \mathcal{L}(F)$ is unambiguous.

The other cases can be handled similarly. For instance, when $Z_{s,t}(p, \rightsquigarrow) = (\rightsquigarrow, q)$ as on the right of Figure 6, the sequence of steps ends with $x_{2n-1} = (\rightsquigarrow, p_{2n-1}, p_{2n}) \in t$ with $n > 0$ and $p_{2n} = q$ and the zigzag run ρ is $\rho_1\rho_2\cdots\rho_{2n-1}$. The SDRTE $\mathsf{ZC}^{F,t}_{E,s}(p, \rightsquigarrow)$ is given by

$$\big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}(x_1)\big) \odot \big(\mathsf{C}_{E,s}(x_2) \cdot \mathsf{C}_{F,t}(x_3)\big) \odot \cdots \cdots \odot \big(\mathsf{C}_{E,s}(x_{2n-2}) \cdot \mathsf{C}_{F,t}(x_{2n-1})\big).$$

The situation is symmetric for $\mathsf{ZC}^{F,t}_{E,s}(p, \subsetneq)$: the sequence starts with a right-right step $x_2 = (\subsetneq, p_2, p_3) \in s$ with $p = p_2$ and we obtain the SDRTE simply by removing the first factor $\big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}(x_1)\big)$ in the Hadamard products above. □

We come back to the definition of the SDRTEs for $\mathsf{C}_{E\cdot F,r}(x)$ with $r \in \mathsf{TrM}$ and $x \in r$. As explained above, the output produced by a run ρ following step x over a word $w = uv$ with $u \in \mathcal{L}(E,s)$, $v \in \mathcal{L}(F,t)$ and $r = st$ consists of an initial part, a zigzag internal part, and a final part. There are four cases depending on the step x.

• $x = (\rightsquigarrow, p, q)$. Either the run ρ stays inside u (zigzag part empty) or there is a zigzag internal part starting with $(p', \rightsquigarrow)$ such that $(\rightarrow, p, p') \in \varphi(u)$ and ending with $(\rightsquigarrow, q')$ such that $(\leftarrow, q', q) \in \varphi(u)$. Thus we define the SDRTE $\mathsf{C}_{E\cdot F,r}(x)$ as

$$\sum_{st=r \mid x \in s} \mathsf{C}_{E,s}(x) \cdot \big(\mathcal{L}(F,t) \triangleright \varepsilon\big) + \sum_{\substack{st=r,\,(p',q') \mid \\ Z_{s,t}(p',\rightsquigarrow)=(\rightsquigarrow,q')}} \big(\mathsf{C}_{E,s}((\rightarrow,p,p')) \cdot (\mathcal{L}(F,t) \triangleright \varepsilon)\big) \odot \mathsf{ZC}^{F,t}_{E,s}(p', \rightsquigarrow) \odot \big(\mathsf{C}_{E,s}((\leftarrow,q',q)) \cdot (\mathcal{L}(F,t) \triangleright \varepsilon)\big)$$

Notice that all Cauchy products are unambiguous since the concatenation $\mathcal{L}(E)\cdot\mathcal{L}(F)$ is unambiguous. The sums are also unambiguous. Indeed, a word $w \in \mathcal{L}(E \cdot F, r)$ has a unique factorization $w = uv$ with $u \in \mathcal{L}(E)$ and $v \in \mathcal{L}(F)$. Hence $s = \varphi(u)$ and $t = \varphi(v)$ are uniquely determined and satisfy $st = r$. Then, either $x \in s$ and w is only in the domain of $\mathsf{C}_{E,s}(x)\cdot\big(\mathcal{L}(F,t)\triangleright\varepsilon\big)$. Or there is a unique p' with $(\rightarrow, p, p') \in s$ and a unique q' with $Z_{s,t}(p', \rightsquigarrow) = (\rightsquigarrow, q')$ and $(\leftarrow, q', q) \in s$. Notice that if $(\rightarrow, p, p') \notin s$ then $\mathsf{C}_{E,s}((\rightarrow, p, p')) = \bot$ and similarly if $(\leftarrow, q', q) \notin s$. Hence we could have added the condition $(\rightarrow, p, p'), (\leftarrow, q', q) \in s$ to the second sum, but do not, to reduce clutter.

• $x = (\rightarrow, p, q)$. Here the run must cross from left to right. Thus we define the SDRTE $\mathsf{C}_{E\cdot F,r}(x)$ as

$$\sum_{\substack{st=r,\,(p',q') \mid \\ Z_{s,t}(p',\rightsquigarrow)=(\subsetneq,q')}} \big(\mathsf{C}_{E,s}((\rightarrow,p,p')) \cdot (\mathcal{L}(F,t) \triangleright \varepsilon)\big) \odot \mathsf{ZC}^{F,t}_{E,s}(p', \rightsquigarrow) \odot \big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}((\rightarrow,q',q))\big)$$

• $x = (\leftarrow, p, q)$. This case is similar, $\mathsf{C}_{E\cdot F,r}(x)$ is

$$\sum_{\substack{st=r,\,(p',q') \mid \\ Z_{s,t}(p',\subsetneq)=(\rightsquigarrow,q')}} \big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}((\leftarrow,p,p'))\big) \odot \mathsf{ZC}^{F,t}_{E,s}(p', \subsetneq) \odot \big(\mathsf{C}_{E,s}((\leftarrow,q',q)) \cdot (\mathcal{L}(F,t) \triangleright \varepsilon)\big)$$

• $x = (\subsetneq, p, q)$. Finally, for right-right runs, $\mathsf{C}_{E\cdot F,r}(x)$ is

$$\sum_{st=r \mid x \in t} (\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}(x) + \sum_{\substack{st=r,\,(p',q') \mid \\ Z_{s,t}(p',\subsetneq)=(\subsetneq,q')}} \big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}((\leftarrow,p,p'))\big) \odot \mathsf{ZC}^{F,t}_{E,s}(p', \subsetneq) \odot \big((\mathcal{L}(E,s) \triangleright \varepsilon) \cdot \mathsf{C}_{F,t}((\rightarrow,q',q))\big)$$

SD-Kleene Star The most interesting case is when $E = F^*$. Let $L = \mathcal{L}(F) \subseteq \Sigma^*$. Since E is a stabilising SD-regular expression, L is an aperiodic prefix code of bounded synchronisation delay, and $X = \varphi(L)$ is k-stabilising for some $k > 0$. Hence, we may apply the results of Section V-B.

By induction, we suppose that we have SDRTEs $\mathsf{C}_{F,s}(x)$ for all s in TrM and steps x. Since $L = \mathcal{L}(F)$ is a code, for each fixed $\ell > 0$, the expression $F^\ell = F \cdot F \cdots F$ is an unambiguous concatenation. Hence, from the proof above for the unambiguous concatenation, we may also assume that we have SDRTEs $\mathsf{C}_{F^\ell,s}(x)$ for all $s \in \mathsf{TrM}$ and steps x. Similarly, we have SDRTEs for $\mathsf{ZC}^{F,t}_{F^k,s}(-)$ and $\mathsf{ZC}^{F^k,t}_{F,s}(-)$. Notice that F^0 is equivalent to ε hence we have $\mathsf{C}_{F^0,s}(x) = \mathsf{C}_{\varepsilon,s}(x)$.

We show how to construct SDRTEs $\mathsf{C}_{E,s}(x)$ for $E = F^*$. There are four cases, which are dealt with below, depending on the step x. We fix some notations common to all four cases. Fix some $w \in \mathcal{L}(E,s) = L^* \cap \varphi^{-1}(s)$ and let $w = u_1 \cdots u_n$ be the unique factorization of w with $n \geq 0$ and $u_i \in L$ for $1 \leq i \leq n$. For a step $x \in s$, we denote by ρ the unique run of $\mathcal{A}$ over w following step x. Again the constructions are illustrated in [14].

• $x = (\rightsquigarrow, p, q) \in s$. The easiest case is for left-left steps. If $n < k$ then the output of ρ is $[\![\mathsf{C}_{F^n,s}(x)]\!](w)$. Notice that here, $\mathsf{C}_{F^0,s}(x) = \bot$ since $x \notin \mathbf{1} = \varphi(\varepsilon)$. Now, if $n \geq k$ then, by Lemma 3, the run ρ stays inside $u_1 \cdots u_k$. We deduce that the output of ρ is $[\![\mathsf{C}_{F^k,s}(x)]\!](u_1 \cdots u_k)$. Therefore, we define

$$\mathsf{C}_{E,s}(x) = \Big(\sum_{n<k} \mathsf{C}_{F^n,s}(x)\Big) + \Big(\mathsf{C}_{F^k,s}(x) \cdot (F^* \triangleright \varepsilon)\Big)$$

Notice that the sums are unambiguous since $L = \mathcal{L}(F)$ is a code. The concatenation $F^k \cdot F^*$ is also unambiguous.

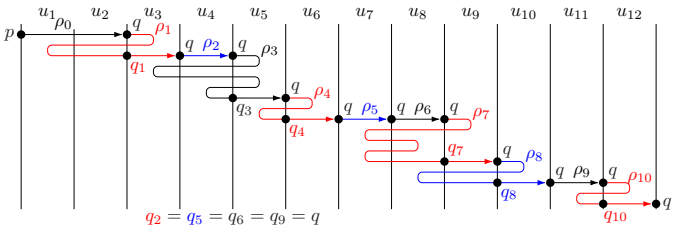


Fig. 7. A left-right run which is 2-forward-progressing.

• $x = (\rightarrow, p, q) \in s$. We turn now to the more interesting left-right steps. Again, if $n < k$ then the output of ρ is $[\![\mathsf{C}_{F^n,s}(x)]\!](w)$. Assume now that $n \geq k$. We apply Lemma 4 to deduce that the run ρ is k-forward-progressing. See Figure 7 for a sample run which is 2-forward-progressing. We split ρ in $\rho_0\rho_1 \cdots \rho_{n-k}$ where ρ_0 is the prefix of ρ going until the first crossing from u_k to u_{k+1}. Then, ρ_1 goes until the first crossing from u_{k+1} to u_{k+2}. Continuing in the same way, for $1 \leq i < n-k$, ρ_i goes until the first crossing from u_{k+i} to u_{k+i+1}. Finally, ρ_{n-k} is the remaining suffix, going until the run exits from w on the right. Since the run ρ is k-forward progressing, we deduce that ρ_i does not go back to $u_1 \cdots u_{i-1}$, hence it stays inside $u_i \cdots u_{i+k}$, starting on the left of u_{i+k} and exiting on the right of u_{i+k}.

Since $X = \varphi(L)$ is k-stabilising, we have $\varphi(u_1 \cdots u_{k+i}) = \varphi(w)$ for all $0 \leq i \leq n-k$. Now, $\rho_0 \cdots \rho_i$ is a run on $u_1 \cdots u_{k+i}$ starting on the left in state p and exiting on the right. Since $\mathcal{A}$ is deterministic and $x = (\rightarrow, p, q) \in \varphi(w) = \varphi(u_1 \cdots u_{k+i})$ we deduce that ρ_i exits on the right of u_{k+i} in state q. In particular, ρ_0 is a run on $u_1 \cdots u_k$ starting on the left in state p and exiting on the right in state q. Moreover, for each $1 \leq i \leq n-k$, ρ_i is the concatenation of an internal zigzag run over $(u_i \cdots u_{i+k-1}, u_{i+k})$ starting with $(q, \leftrightsquigarrow)$ ending with $(\subsetneq, q_i) = Z_{s',s''}(q, \leftrightsquigarrow)$ where $s' = \varphi(u_i \cdots u_{i+k-1})$, $s'' = \varphi(u_{i+k})$ and a $(\rightarrow, q_i, q)$ run over u_{i+k}.

Let v_i be the output produced by ρ_i for $0 \leq i \leq n-k$. Then, using Lemma 7, the productions v_i with $0 < i \leq n-k$ are given by the SDRTE f defined as

$$f = \sum_{\substack{s',s'',q' \mid \\ (\subsetneq,q') = Z_{s',s''}(q,\leftrightsquigarrow)}} \mathsf{ZC}^{F,s''}_{F^k,s'}(q, \leftrightsquigarrow) \odot \big((\mathcal{L}(F^k, s') \triangleright \varepsilon) \cdot \mathsf{C}_{F,s''}((\rightarrow, q', q))\big)$$

Then the product $v_1 \cdots v_{n-k}$ is produced by the $(k+1)$-chained Kleene-star $[L, f]^{(k+1)\star}(w)$. From the above discussion, we also deduce that $v_0 = [\![\mathsf{C}_{F^k,s}(x)]\!](u_1 \cdots u_k)$. Therefore, we define $\mathsf{C}_{E,s}(x) =$

$$\Big(\sum_{n<k} \mathsf{C}_{F^n,s}(x)\Big) + \Big(\big(\mathsf{C}_{F^k,s}(x) \cdot (F^* \triangleright \varepsilon)\big) \odot [F, f]^{(k+1)\star}\Big)$$

• $x = (\leftarrow, p, q) \in s$. Though slightly more complicated, the case of right-left runs is symmetric to the case of left-right runs. The proof is in [14].

• $x = (\subsetneq, p, q) \in s$. As explained in Lemma 6 and Figure 5, a right-right run combines a right-left run, a zigzag, and a left-right run. The detailed proof is in [14]. □

We conclude the section by showing how to construct SDRTEs equivalent to 2DFTs.

Theorem V.3. *Let $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \gamma, q_0, F)$ be an aperiodic 2DFT. We can construct an equivalent SDRTE $C_\mathcal{A}$ over alphabet Σ with $\mathsf{dom}([\![C_\mathcal{A}]\!]) = \mathsf{dom}([\![\mathcal{A}]\!])$ and $[\![\mathcal{A}]\!](w) = [\![C_\mathcal{A}]\!](w)$ for all $w \in \mathsf{dom}([\![\mathcal{A}]\!])$.*

Proof. We first construct below an SDRTE $C'_\mathcal{A}$ with $\mathsf{dom}([\![C'_\mathcal{A}]\!]) = \vdash\mathsf{dom}([\![\mathcal{A}]\!])\dashv$ and such that $[\![\mathcal{A}]\!](w) = [\![C'_\mathcal{A}]\!](\vdash w\dashv)$ for all $w \in \mathsf{dom}([\![\mathcal{A}]\!])$. Then, we obtain $C''_\mathcal{A}$ using Proposition 1 by $C''_\mathcal{A} = (\vdash^{-1} C'_\mathcal{A})\dashv^{-1}$.

Finally, we get rid of lingering endmarkers in $C''_\mathcal{A}$ using Lemma 2 to obtain $C_\mathcal{A}$ as the projection of $C''_\mathcal{A}$ on Σ^*.

Let $\varphi\colon (\Sigma \uplus \{\vdash, \dashv\})^* \to \mathsf{TrM}$ be the canonical surjective morphism to the transition monoid of $\mathcal{A}$. Since $\mathcal{A}$ is aperiodic, the monoid TrM is also aperiodic. We can apply Theorem III.2 to the restriction of φ to Σ^*: for each $s \in \mathsf{TrM}$, we get an unambiguous, stabilising, SD-regular expression E_s with $\mathcal{L}(E_s) = \varphi^{-1}(s) \cap \Sigma^*$. Let $E = \vdash \cdot (\bigcup_{s \in \mathsf{TrM}} E_s)$ which is an unambiguous, stabilising, SD-regular expression with $\mathcal{L}(E) = \vdash\Sigma^*$. Applying Theorem V.2, for each monoid element $s \in \mathsf{TrM}$ and each step $x \in \{\rightarrow, \leftrightsquigarrow, \subsetneq, \leftarrow\} \times Q^2$, we construct the corresponding SDRTE $\mathsf{C}_{E,s}(x)$. We also apply Lemma 7 and construct for each state $p \in Q$ an SDRTE $\mathsf{ZC}^{\dashv,t}_{E,s}(p, \leftrightsquigarrow)$ where $t = \varphi(\dashv)$.

Finally, we define

$$C'_\mathcal{A} = \sum_{\substack{s,p,q \mid q \in F \\ (\rightarrow,q_0,p) \in s \\ Z_{s,t}(p,\leftrightsquigarrow) = (\subsetneq,q)}} \big(\mathsf{C}_{E,s}((\rightarrow, q_0, p)) \cdot (\dashv \triangleright \varepsilon)\big) \odot \mathsf{ZC}^{\dashv,t}_{E,s}(p, \leftrightsquigarrow)$$

We can easily check that $C'_\mathcal{A}$ satisfies the requirements above. □

One can find in [14] an application of the construction to our running example.

VI. Adding composition

Since SDRTEs are used to define functions over words, it seems natural to consider the composition of functions, as it is an easy to understand but powerful operator. In this section, we discuss other formalisms using composition as a basic operator, and having the same expressive power as SDRTEs.

Theorem I.1 gives the equivalence between SDRTEs and aperiodic two-way transducers, the latter being known to be closed under composition. Hence, adding composition to SDRTEs does not add expressiveness, while allowing for easier modelisation of transformations.

Moreover, we prove that, should we add composition of functions, then we can replace the k-chained star operator and its reverse by the simpler 1-star $[L, f]^{1\star}$ and its reverse, which in particular are one-way (left-to-right or right-to-left) operator when f is also one-way.

Finally, we prove that we can furthermore get rid of the reverse operators as well as the Hadamard product by adding two basic functions: *reverse* and *duplicate*. The reverse function is straightforward as it reverses its input. The duplicate

function is parameterised by a symbol, say $\#$, duplicates its input inserting $\#$ between the two copies: $\mathsf{dup}_{\#}(u) = u\#u$.

Theorem VI.1. *The following families of expressions have the same expressive power:*

1) *SDRTEs,*
2) *SDRTEs with composition of functions,*
3) *SDRTEs with composition and 1-star only.*
4) *Expressions with simple functions, unambiguous sum, Cauchy product,* 1*-star, duplicate, reverse, composition.*

Proof. It is trivial that $3 \subseteq 2$ as 3 is obtained as a syntaxical restriction of 2. Although it is not needed in our proof, $1 \subseteq 2$ holds for the same reason. Now, thanks to Theorem I.1, we know that SDRTEs are equivalent to aperiodic 2DFTs that are closed under composition. Hence, composition does not add expressive power and we have $2 \subseteq 1$.

To prove that $4 \subseteq 3$, we simply have to prove that the duplicate and reverse functions can be expressed with SDRTEs using only the 1-star operator and its reverse. The duplicate function definition relies on the Hadamard and is given by the expression: $\mathsf{dup}_{\#} = (\mathsf{id}_{\Sigma^*} \cdot (\varepsilon \triangleright \#)) \odot \mathsf{id}_{\Sigma^*}$

where id_{Σ^*} is the identity function and can be written as $[\Sigma, \mathsf{id}_\Sigma]^{1\star}$ where $\mathsf{id}_\Sigma = \sum_{a\in\Sigma} a \triangleright a$. The reverse function is also easy to define using the 1-star reverse: $\mathsf{rev} = [\Sigma, \mathsf{id}_\Sigma]_r^{1\star}$

To prove the last inclusion $1 \subseteq 4$, we need to express the Hadamard product and the (reverse) k-chained star, using duplicate, reverse and composition.

The Hadamard product $f \odot g$ is easy to define using $\mathsf{dup}_{\#}$ where $\#$ is a fresh marker:

$$f \odot g = (f \cdot (\# \triangleright \varepsilon) \cdot g) \circ \mathsf{dup}_{\#} .$$

We show now how to reduce k-star to 1-star using duplicate and composition. The proof is by induction on k. When $k = 1$ there is nothing to do. Assume that $k > 1$. We show how to express $[L, f]^{k\star}$ using $(k-1)$-star, 1-star and duplicate. The main idea is to use composition to mark each factor in L in order to duplicate them, then use a $(k-1)$-star to have a reach of k factors of L (with some redundant information), and lastly use composition to prune the input to a form suitable to finally apply f.

More formally, let $\#$ and \$ be two fresh markers and define

$$f_1 = [L, \mathsf{dup}_{\$} \cdot (\varepsilon \triangleright \#)]^{1\star}$$

with domain L^* and, when applied to a word $u = u_1 \cdots u_n$ with $u_i \in L$, produces

$$u_1\$u_1\#u_2\$u_2\#u_3\$u_3\# \cdots u_{n-1}\$u_{n-1}\#u_n\$u_n\#$$

in $\{\varepsilon\} \cup \Sigma^*\$(\Sigma^*\#\Sigma^*\$)^*\Sigma^*\#$. Notice that $\Sigma^*\#\Sigma^*\$$ is a 1-SD prefix code and that taking $k-1$ consecutive factors from this language allows us to have a reach of k factors of L.

Then we define the function g

$$g = \big(\mathsf{id}_{\Sigma^*} \cdot (\#\Sigma^*\$ \triangleright \varepsilon)\big)^{k-2} \cdot \big(\mathsf{id}_{\Sigma^*} \cdot (\# \triangleright \varepsilon) \cdot \mathsf{id}_{\Sigma^*} \cdot (\$ \triangleright \varepsilon)\big)$$

with domain $(\Sigma^*\#\Sigma^*\$)^{k-1}$, it produces $v_1v_2 \cdots v_{k-1}v'_{k-1}$ when applied to $v_1\#v'_1\$v_2\#v'_2\$ \cdots v_{k-1}\#v'_{k-1}\$$. In particular, $g(u_{i+1}\#u_{i+2}\$u_{i+2}\#u_{i+3}\$ \cdots u_{i+k-1}\#u_{i+k}\$)$ gives $u_{i+1} \cdots u_{i+k}$. Finally, we have $[L, f]^{k\star} =$

$$\big((\varepsilon \triangleright \varepsilon) + (\Sigma^*\$ \triangleright \varepsilon) \cdot [\Sigma^*\#\Sigma^*\$, f \circ g]^{(k-1)\star} \cdot (\Sigma^*\# \triangleright \varepsilon)\big) \circ f_1 .$$

The reverse k-star $[L, f]_r^{k\star}$ is not expressed in a straightforward fashion using reverse composed with k-star, because while reverse applies on all the input, the reverse k-star swaps the applications of function f while keeping the function f itself untouched. In order to express it, we reverse a k-star operator not on f, but on f reversed. The result is that the applications of f are reversed twice, thus preserving them. Formally, we have: $[L, f]_r^{k\star} = \mathsf{rev} \circ [L, \mathsf{rev} \circ f]^{k\star}$. □

VII. Conclusion

We conclude with some interesting avenues for future work, arising from the open questions based on this paper.

We begin with complexity questions, and then move on to other directions for future work. The complexity of our procedure, especially when going from the declarative language SDRTE to the operational 2DFT machine, is open. This part relies heavily on the composition of 2DFTs which incurs at least one exponential blowup in the state space. A possibility to reduce the complexity incurred during composition, is to obtain *reversible* 2FT (2RFT) for each of the intermediate functions used in the composition. 2RFTs are a class of 2DFTs which are both deterministic and co-deterministic, and were introduced in [20], where they prove that composition of 2RFTs results in a 2RFT with polynomially many states in the number of states of the input transducers. Provided that the composition of 2RFTs preserves aperiodicity, if we could produce 2RFTs in our procedures in section V-A, then we would construct a 2RFT which is polynomial in the size of the SDRTE. Another open question is the efficiency of evaluation, i.e., given an SDRTE and an input word, what is the time complexity of computing the corresponding output. This is crucial for an implementation, along the lines of DReX [21].

Yet another direction is to extend our result to transformations over infinite words. While Perrin [22] generalized the SF=AP result of Schützenberger to infinite words in the mid 1980s, Diekert and Kufleitner [16], [17] generalized recently Schützenberger's SD=AP result to infinite words. Based on this extended SD=AP, one could check how to adapt our proof to the setting of transformations over infinite words. Finally, a long standing open problem in the theory of transformations is to decide if a function given by a 2DFT is realizable by an aperiodic one. This question has been solved in the one-way case [23], or in the case when we have *origin* information [24], but the general case remains open. We believe that our characterisation of stabilising runs provided in Section V-B could lead to forbidden pattern criteria to decide this open problem.

References

[1] J. Engelfriet and H. J. Hoogeboom, "MSO definable string transductions and two-way finite-state transducers," *ACM Transactions on Computational Logic*, vol. 2, no. 2, pp. 216–254, 2001. [Online]. Available: http://dx.doi.org/10.1145/371316.371512

[2] B. Courcelle, "Monadic second-order definable graph transductions: a survey [see MR1251992 (94f:68009)]," *Theoret. Comput. Sci.*, vol. 126, no. 1, pp. 53–75, 1994, seventeenth Colloquium on Trees in Algebra and Programming (CAAP '92) and European Symposium on Programming (ESOP) (Rennes, 1992). [Online]. Available: http://dx.doi.org/10.1016/0304-3975(94)90268-2

[3] R. Alur, A. Freilich, and M. Raghothaman, "Regular combinators for string transformations," in *Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, T. A. Henzinger and D. Miller, Eds. ACM, 2014, pp. 9:1–9:10.

[4] N. Baudru and P.-A. Reynier, "From two-way transducers to regular function expressions," in *22nd International Conference on Developments in Language Theory, DLT 2018*, ser. Lecture Notes in Computer Science, M. Hoshi and S. Seki, Eds., vol. 11088. Springer, 2018, pp. 96–108.

[5] V. Dave, P. Gastin, and S. N. Krishna, "Regular Transducer Expressions for Regular Transformations," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic In Computer Science (LICS'18)*, M. Hofmann, A. Dawar, and E. Grädel, Eds. Oxford, UK: ACM Press, Jul. 2018, pp. 315–324.

[6] R. Alur and P. Černý, "Expressiveness of streaming string transducers," in *30th International Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010*, ser. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2010, vol. 8, pp. 1–12.

[7] M.-P. Schützenberger, "On finite monoids having only trivial subgroups," *Information and Control*, vol. 8, no. 2, pp. 190–194, 1965.

[8] R. McNaughton and S. Papert, *Counter-Free Automata*. Cambridge, Mass.: The MIT Press, 1971.

[9] O. Carton and L. Dartois, "Aperiodic two-way transducers and fo-transductions," in *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, ser. LIPIcs, S. Kreutzer, Ed., vol. 41. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 160–174. [Online]. Available: https://doi.org/10.4230/LIPIcs.CSL.2015.160

[10] E. Filiot, S. N. Krishna, and A. Trivedi, "First-order definable string transformations," in *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, ser. LIPIcs, V. Raman and S. P. Suresh, Eds., vol. 29. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014, pp. 147–159. [Online]. Available: http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2014.147

[11] M.-P. Schützenberger, "Sur certaines opérations de fermeture dans les langages rationnels," in *Symposia Mathematica, Vol. XV (Convegno di Informatica Teorica, INDAM, Roma, 1973)*. Academic Press, 1975, pp. 245–253.

[12] P. Gastin, "Modular descriptions of regular functions," in *Algebraic Informatics - 8th International Conference, CAI 2019, Niš, Serbia, June 30 - July 4, 2019, Proceedings*, 2019, pp. 3–9. [Online]. Available: https://doi.org/10.1007/978-3-030-21363-3_1

[13] M. Bojanczyk, L. Daviaud, and S. N. Krishna, "Regular and first-order list functions," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, 2018, pp. 125–134. [Online]. Available: https://doi.org/10.1145/3209108.3209163

[14] L. Dartois, P. Gastin, and S. N. Krishna, "Sd-regular transducer expressions for aperiodic transformations," *CoRR*, vol. abs/2101.07130, 2021. [Online]. Available: https://arxiv.org/abs/2101.07130

[15] J. R. Büchi, "Weak second-order arithmetic and finite automata," *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 6, pp. 66–92, 1960.

[16] V. Diekert and M. Kufleitner, "Bounded synchronization delay in omega-rational expressions," in *Computer Science - Theory and Applications - 7th International Computer Science Symposium in Russia, CSR 2012, Nizhny Novgorod, Russia, July 3-7, 2012. Proceedings*, 2012, pp. 89–98. [Online]. Available: https://doi.org/10.1007/978-3-642-30642-6_10

[17] ——, "Omega-rational expressions with bounded synchronization delay," *Theory of Computing Systems*, vol. 56, no. 4, pp. 686–696, 2015.

[18] ——, "A survey on the local divisor technique," *Theoretical Computer Science*, vol. 610, pp. 13–23, Jan 2016.

[19] D. Perrin and J.-E. Pin, *Infinite Words: Automata, Semigroups, Logic and Games*. Elsevier, 2004, vol. 141.

[20] L. Dartois, P. Fournier, I. Jecker, and N. Lhote, "On reversible transducers," in *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, ser. LIPIcs, I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, Eds., vol. 80. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 113:1–113:12. [Online]. Available: https://doi.org/10.4230/LIPIcs.ICALP.2017.113

[21] R. Alur, L. D'Antoni, and M. Raghothaman, "Drex: A declarative language for efficiently evaluating regular string transformations," in *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, 2015, pp. 125–137. [Online]. Available: https://doi.org/10.1145/2676726.2676981

[22] D. Perrin, "Recent results on automata and infinite words," in *Mathematical Foundations of Computer Science 1984, Praha, Czechoslovakia, September 3-7, 1984, Proceedings*, 1984, pp. 134–148. [Online]. Available: https://doi.org/10.1007/BFb0030294

[23] E. Filiot, O. Gauwin, and N. Lhote, "Logical and algebraic characterizations of rational transductions," *Log. Methods Comput. Sci.*, vol. 15, no. 4, 2019. [Online]. Available: https://doi.org/10.23638/LMCS-15(4:16)2019

[24] M. Bojanczyk, "Transducers with origin information," in *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, 2014, pp. 26–37. [Online]. Available: https://doi.org/10.1007/978-3-662-43951-7_3