Untwisting two-way transducers in elementary time

Untwisting two-way transducers in elementary time

Félix Baschenis*, Olivier Gauwin*, Anca Muscholl* and Gabriele Puppis*
*Université de Bordeaux, LaBRI, CNRS

***Abstract*—Functional transductions realized by two-way transducers (equivalently, by streaming transducers and by MSO transductions) are the natural and standard notion of "regular" mappings from words to words. It was shown recently (LICS'13) that it is decidable if such a transduction can be implemented by some one-way transducer, but the given algorithm has non-elementary complexity. We provide an algorithm of different flavor solving the above question, that has double exponential space complexity. We further apply our technique to decide whether the transduction realized by a two-way transducer can be implemented by a sweeping transducer, with either known or unknown number of passes.**

1. Introduction

Since the early times of computer science, transducers have been identified as a fundamental notion of computation, where one is interested how objects can be transformed into each other. Numerous fields of computer science are ultimately concerned with transformations, ranging from databases to image processing, and an important issue is to perform transformations with low costs, whenever possible.

The most basic form of transformers are devices that process an input and produce outputs during the processing, using finite memory. Such devices are called finite-state transducers. Word-to-word finite-state transducers were considered in very early work in formal language theory [1], [11], [22], and it was soon clear that they are much more challenging than finite-state word acceptors - the classical finite-state automata. One essential difference between transducers and automata over words is that the capability to process the input in both directions strictly increases the expressive power in the case of transducers, whereas this does not for automata [20], [23]. In other words, two-way word transducers are strictly more expressive than one-way word transducers.

We consider in this paper functional transducers - nondeterministic transducers that compute functions from words to words (also called one-valued in the literature). Two-way functional transducers capture very nicely the notion of regularity in this setting. Regular word functions, i.e. functions computed by functional two-way transducers, inherit many of the characterizations and algorithmic properties of the robust class of regular languages. Engelfriet and Hoogeboom [12] showed that monadic second-order definable graph transductions, restricted to words, are equivalent to two-way transducers — this justifies the notation "regular" word functions, in the spirit of classical results in automata theory and logic by Büchi, Elgot, Rabin and others. Recently, Alur and Cerný [2] proposed an enhanced version of one-way transducers called streaming transducers, and showed that they are equivalent to the two previous models. A streaming transducer processes the input word from left to right, and stores (partial) output words in finitely many, write-only registers.

Two-way transducers raise challenging questions about resource requirements. One crucial resource is the number of times the transducer needs to re-process the input word. In particular, the case where the input can be processed in a single pass, from left to right, is very attractive as it corresponds to the setting of *streaming*, where the (possibly very large) inputs do not need to be stored in order to be processed. Recently, it was shown in [14] that it is decidable whether the transduction defined by a functional two-way transducer can be implemented by a one-way transducer. However, the decision procedure of [14] has non-elementary complexity, and it is natural to ask whether one can do better. We gave in [3], [4] an exponential space algorithm in the special case of *sweeping* transducers: head reversals are only allowed at the extremities of the input. However, sweeping transducers are known to be strictly less expressive than two-way transducers.

In this paper we provide an algorithm of elementary complexity for deciding whether the transduction defined by a functional two-way transducer can be implemented by a one-way transducer: the decision algorithm has double exponential space complexity, and an equivalent one-way transducer (if it exists) of triple exponential size can be constructed. The known lower bound [3] is double exponential size. We also adapt our techniques to characterize sweeping transducers within the class of two-way transducers.

Related work. Besides the papers mentioned above, there are several recent results around the expressivity and the resources of two-way transducers, or equivalently, streaming transducers. First-order definable transductions were shown to be equivalent to transductions defined by aperiodic streaming transducers [15] and to aperiodic two-way transducers [7]. An effective characterization of aperiodicity for one-way transducers was obtained in [13].

In [4], [10] the minimization of the number of registers of deterministic streaming transducers, resp., passes of functional sweeping transducers, was shown to be decidable. An algebraic characterization of (not necessarily

This work was partially supported by the projects ExStream (ANR-13-JS02-0010) and DeLTA (ANR-16-CE40-0007).

978-1-5090-3018-7/17/$31.00

functional) two-way transducers over unary alphabets was provided in [8]. It was shown that in this case sweeping transducers have the same expressivity. The expressivity of non-deterministic input-unary or output-unary two-way transducers was investigated in [17].

Overview. Section 2 introduces basic notations for two-way transducers, and Section 3 states the main result. Section 4 is devoted to the effect of pumping on outputs, and Section 5 introduces the main tool for our characterization. Section 6 handles the construction of an equivalent one-way transducer. Finally, Section 7 describes a procedure to decide whether a functional transducer is equivalent to a sweeping transducer. Missing proofs can be found in the extended version [5].

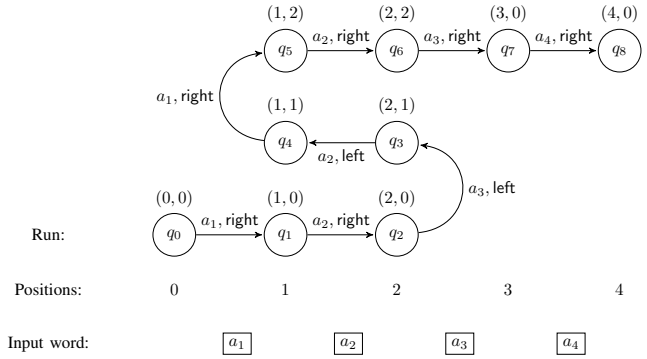


Figure 1. Graphical presentation of a run by means of crossing sequences.

2. Preliminaries

Two-way automata and transducers. We start with some basic notations and definitions for two-way automata (resp., transducers). We assume that every input word $u = a_1 \cdots a_n$ has two special delimiting symbols $a_1 = \vdash$ and $a_n = \dashv$ that do not occur elsewhere: $a_i \notin \{\vdash, \dashv\}$ for all $i = 2, \ldots, n-1$.

A *two-way automaton* $\mathcal{A} = \langle Q, \Sigma, \vdash, \dashv, \delta, q_0, F \rangle$ has a finite state set Q, finite input alphabet Σ, transition relation $\delta \subseteq Q \times (\Sigma \cup \{\vdash, \dashv\}) \times Q \times \{\mathsf{left}, \mathsf{right}\}$, initial state $q_0 \in Q$, and set of final states $F \subseteq Q$. By convention, left transitions on $\vdash$ are not allowed. A *configuration* of $\mathcal{A}$ has the form $u\,q\,v$, with $uv \in \{\vdash\} \cdot \Sigma^* \cdot \{\dashv\}$ and $q \in Q$. A configuration $u\,q\,v$ represents the situation where the current state of $\mathcal{A}$ is q and its head reads the first symbol of v (on input uv). If $(q, a, q', \mathsf{right}) \in \delta$, then there is a transition from any configuration of the form $u\,q\,av$ to the configuration $ua\,q'\,v$, which we denote $u\,q\,av \xrightarrow{a,\mathsf{right}} ua\,q'\,v$. Similarly, if $(q, a, q', \mathsf{left}) \in \delta$, then there is a transition from any configuration of the form $ub\,q\,av$ to the configuration $u\,q'\,bav$, denoted as $ub\,q\,av \xrightarrow{a,\mathsf{left}} u\,q'\,bav$. A *run* on w is a sequence of transitions. It is *successful* if it starts in the initial configuration $q_0\,w$ and ends in a configuration $w\,q$ with $q \in F$ — note that this latter configuration does not allow additional transitions. The *language* of $\mathcal{A}$ is the set of input words that admit a successful run of $\mathcal{A}$.

The definition of *two-way transducers* is similar to that of two-way automata, with the only difference that now there is an additional output alphabet Γ and the transition relation is a finite subset of $Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \Gamma^* \times Q \times \{\mathsf{left}, \mathsf{right}\}$, which associates an output over Γ with each transition of the underlying two-way automaton. Formally, given a two-way transducer $\mathcal{T} = \langle Q, \Sigma, \vdash, \dashv, \Gamma, \delta, q_0, F \rangle$, we have a transition of the form $ub\,q\,av \xrightarrow{a,d|w} u'\,q'\,v'$, outputting w, whenever $(q, a, w, q', d) \in \delta$ and either $u' = uba$, $v' = v$ or $u' = u$, $v' = bav$, depending on whether $d = \mathsf{right}$ or $d = \mathsf{left}$. The *output* associated with a run $\rho = u_1\,q_1\,v_1 \xrightarrow{a_1,d_1|w_1} \cdots \xrightarrow{a_n,d_n|w_n} u_{n+1}\,q_{n+1}\,v_{n+1}$ of $\mathcal{T}$ is the word $\mathsf{out}(\rho) = w_1 \cdots w_n$. A transducer $\mathcal{T}$ defines a relation consisting of all pairs (u, w) such that $w = \mathsf{out}(\rho)$, for some successful run ρ on u.

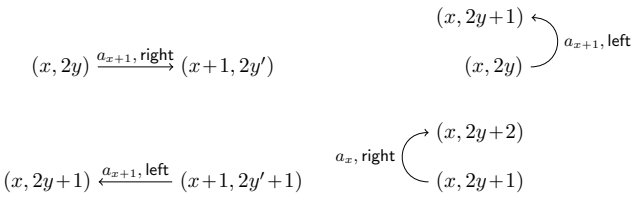
The *domain* of $\mathcal{T}$, denoted $\mathsf{dom}(\mathcal{T})$, is the set of input words that have a successful run. For transducers $\mathcal{T}, \mathcal{T}'$, we write $\mathcal{T}' \subseteq \mathcal{T}$ to mean that $\mathsf{dom}(\mathcal{T}') \subseteq \mathsf{dom}(\mathcal{T})$ and the transductions computed by $\mathcal{T}, \mathcal{T}'$ coincide on $\mathsf{dom}(\mathcal{T}')$.

We say that $\mathcal{T}$ is *functional* if for each input u, at most one output w can be produced by any possible successful run on u. Finally, we say that $\mathcal{T}$ is *one-way* if it does not have transition rules of the form $(q, a, w, q', \mathsf{left})$.

Crossing sequences. The first basic notion is that of crossing sequence. We follow the convenient presentation from [18], which appeals to a graphical representation of runs of a two-way transducer where each configuration is seen as point (location) in a two-dimensional space. Let $u = a_1 \cdots a_n$ be an input word (recall that $a_1 = \vdash$ and $a_n = \dashv$) and let ρ be a run of a two-way automaton (or transducer) $\mathcal{T}$ on u. The *positions* of ρ are the numbers from 0 to n, corresponding to "cuts" between two consecutive letters of the input. For example, position 0 is just before the first letter a_1, position n is just after the last letter a_n, and any other position x, with $1 \leqslant x < n$, is between the letters a_x and a_{x+1}. We say that a transition $u\,q\,v \xrightarrow{a,d} u'\,q'\,v'$ of ρ *crosses position* x if either $d = \mathsf{right}$ and $|u| = x$, or $d = \mathsf{left}$ and $|u'| = x$. A *location* of ρ is any pair (x, y) for which there are at least $y + 1$ transitions in ρ crossing position x; the component y of a location is called *level*. Each location is associated a state. Formally, we say that q is the *state at location* $\ell = (x, y)$ in ρ, and we denote this by writing $\rho(\ell) = q$, if the $(y+1)$-th transition that crosses x ends up in state q. The *crossing sequence* at position x of ρ is the tuple $\rho|x = (q_0, \ldots, q_h)$, where the q_y's are all the states at locations of the form (x, y), for $y = 0, \ldots, h$.

As suggested by Fig. 1, any run can be represented as an annotated path between locations. For example, if a location (x, y) is reached by a rightward transition, then the head of the automaton has read the symbol a_x; if it is reached by a leftward transition, then the head has read the symbol a_{x+1}. Note that in a successful run ρ every crossing sequence has odd length and every rightward (resp. leftward) transition reaches a location with even (resp. odd) level. We identify four types of transitions between locations, depending on the parities of the levels, see Fig. 1. Hereafter, we will identify runs with the corresponding annotated paths

between locations.



It is also convenient to define a total order $\trianglelefteq$ on the locations of a run ρ by letting $\ell_1 \trianglelefteq \ell_2$ if ℓ_2 is reachable from ℓ_1 by following the path described by ρ — the order $\trianglelefteq$ on locations is called *run order*. Given two locations $\ell_1 \trianglelefteq \ell_2$ of a run ρ, we write $\rho[\ell_1, \ell_2]$ for the factor of the run that starts in ℓ_1 and ends in ℓ_2. Note that the latter is also a run and hence the notation $\mathsf{out}\big(\rho[\ell_1, \ell_2]\big)$ is permitted. Two runs ρ_1, ρ_2 can be concatenated, provided that ρ_1 ends in location (x, y), ρ_2 starts in location (x, y'), such that $y' = y \pmod 2$ and (x, y), (x, y') are labelled by the same state. We denote by $\rho_1\rho_2$ the run resulting from concatenating ρ_1 with ρ_2. Clearly, we have $\rho[\ell_1, \ell_2]\ \rho[\ell_2, \ell_3] = \rho[\ell_1, \ell_3]$ for all locations $\ell_1 \trianglelefteq \ell_2 \trianglelefteq \ell_3$.

Normalization. Without loss of generality, we will assume that successful runs of functional transducers are *normalized*, meaning that they never visit two locations with the same position, the same state, and both either at even or at odd level. Indeed, if this were not the case, say if a successful run ρ visited two locations $\ell_1 = (x, y)$ and $\ell_2 = (x, y')$ such that $\rho(\ell_1) = \rho(\ell_2)$ and y, y' are both even or both odd, then the output produced by ρ between ℓ_1 and ℓ_2 should be empty, as otherwise by repeating the factor $\rho[\ell_1, \ell_2]$ of ρ we could obtain successful runs that produces different outputs on the same input, thus contradicting the assumption that the transducer is functional. Now that we know that the output of ρ produced between ℓ_1 and ℓ_2 is empty, we could drop the factor $\rho[\ell_1, \ell_2]$, thus obtaining a successful run with the same output. So in every normalized successful run, crossing sequences are at most $2|Q|-1$ long.

We define $h_{\mathsf{max}} = 2|Q|-1$. Moreover, by c_{max} we denote the maximal length of the output of a transition.

3. Two-way transducers versus one-way transducers

In this section we state our main result, which is the existence of an elementary algorithm for checking whether a two-way transducer is equivalent to some one-way transducer. We call such transducers *one-way definable*. Before doing so, we discuss a few examples.

Example 1. We consider two-way transducers that accept any input u from a given regular language R and output the word $u\,u$. We will argue how, depending on R, these transducers may or may not be one-way definable.

1) If $R = (a + b)^*$ there is no equivalent one-way transducer, as the output language is not regular. If R is finite, then the transduction mapping $u \in R$ to $u\,u$ can be implemented by a one-way transducer that guesses u (this requires as many states as the size of R), checks the input, and outputs two copies of the guessed word.
2) A special case of transduction with finite domain is given by $R_n = \{a_0\, w_0 \cdots a_{2^n-1}\, w_{2^n-1} : a_0, \ldots, a_{2^n-1} \in \{a, b\}\}$, where $n \in \mathbb{N}$ and each w_i is the binary encoding of the counter $i = 0, \ldots, 2^n - 1$. It is easy to see (cf. Proposition 15 [3]) that the transduction mapping $u \in R_n$ to $u\,u$ can be implemented by a two-way transducer with quadratically many states w.r.t. n, while every equivalent one-way transducer has at least 2^{2^n} states, since it needs to guess a word of length 2^n.
3) Consider now the periodic language $R = (abc)^*$. The function that maps $u \in R$ to $u\,u$ can be easily implemented by a one-way transducer: it suffices to output two letters (i.e., ab, ca, bc, in turn) for each input letter, while checking that the input is in R.

Example 2. We consider a slightly more complicated transduction that is defined on input words of the form $u_1 \,\#\, \ldots \,\#\, u_n$, where each factor u_i is over the alphabet $\Sigma = \{a, b, c\}$. The output is $w_1 \# \ldots \# w_n$, where each w_i is either $u_i\, u_i$ or u_i, depending on whether or not $u_i \in (abc)^*$ and u_{i+1} has even length (with $u_{n+1} = \varepsilon$).

The obvious way to implement the transduction is by means of a two-way transducer that performs multiple passes on the factors of the input: a first left-to-right pass is performed on $u_i \# u_{i+1}$ to produce the first copy of u_i and to check whether $u_i \in (abc)^*$ and $|u_{i+1}|$ is even; if so, a second pass on u_i is performed to produce another copy of u_i.

The transduction can also be implemented by a one-way transducer: when entering a factor u_i, the transducer guesses whether or not $u_i \in (abc)^*$ and $|u_{i+1}|$ is even; depending on this it outputs either $(abc\,abc)^{\frac{|u_i|}{3}}$ or u_i, and checks that the guess is correct.

We state now the main result of our paper:

Theorem 3. There is an algorithm that from a functional two-way transducer $\mathcal{T}$ constructs in triple exponential time a *one-way* transducer $\mathcal{T}'$ with the following properties:

- $\mathcal{T}' \subseteq \mathcal{T}$,
- $\mathsf{dom}(\mathcal{T}) = \mathsf{dom}(\mathcal{T}')$ iff $\mathcal{T}$ is one-way definable.

Moreover, the second property above can be checked in double exponential space w.r.t. $|\mathcal{T}|$.

A similar characterization for a much more restricted class of transducers (sweeping transducers) appeared in [3]. The proof of Theorem 3, however, is more technical, as it requires a better understanding of the structure of the runs of two-way transducers and a non-trivial generalization of the combinatorial arguments from [3].

The proof of the above theorem spans along the next three sections. In Section 4, we present the basic concepts for reasoning on runs of two-way automata. This includes the definition of a finite semigroup for describing the shapes of two-way runs, as well as Ramsey-type arguments that

are used to bound the length of the outputs produced by pieces of runs without loops. In Section 5 we provide the main combinatorial arguments for characterizing one-way definability. The crucial notion will be that of *inversion*, that captures behaviours of the two-way transducer that are problematic for one-way definability. Finally, in Section 6 we exploit the combinatorial results and the Ramsey-type arguments to derive the existence of suitable decompositions of runs that lead to the construction of equivalent one-way transducers.

4. Untangling runs of two-way transducers

This section is devoted to the structure of runs of two-way transducers. Whereas the classical transformation of two-way automata into one-way automata based on crossing sequences is rather simple, for transducers we need a more detailed analysis of runs because of the additional outputs. In a nutshell, one-way definability is related to periodicities (with bounded periods) in the output, and these periodicities are generated by loops in the run. We will actually work with so called idempotent loops, that generate periodicities in the output in a "nice" way. We will derive the existence of idempotent loops with bounded outputs using Ramsey-based arguments.

We fix throughout the paper a functional two-way transducer $\mathcal{T}$, an input word u, and a successful run ρ of $\mathcal{T}$ on u. We assume that ρ is normalized, i.e., every state occurs at most once in each crossing sequence of ρ at levels of a given parity.

For simplicity, we denote by ω the length of the input word u. We will consider *intervals of positions* of the form $I = [x_1, x_2]$, with $0 \leqslant x_1 < x_2 \leqslant \omega$. The *containment relation* $\subseteq$ on intervals is defined by $[x_3, x_4] \subseteq [x_1, x_2]$ if $x_1 \leqslant x_3 < x_4 \leqslant x_2$.

Factors, flows, and effects. A factor of a run ρ is a contiguous subsequence of ρ. A factor *intercepted* by an interval $I = [x_1, x_2]$ is a maximal factor of ρ that visits only positions $x \in I$, and never uses a left transition from position x_1 or a right transition from position x_2.

Fig. 2 on the right gives an example of an interval I that intercepts the factors $\alpha, \beta, \gamma, \delta, \zeta$. The numbers that annotate the endpoints of the factors represent their levels.

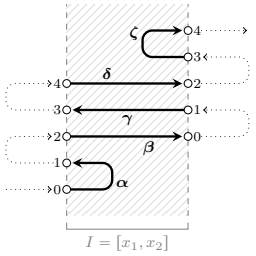


Figure 2. Intercepted factors.

Every factor α intercepted by an interval $I = [x_1, x_2]$ is of one of the four types below, depending on its first location (x, y) and its last location (x', y'):

- α is an LL-factor if $x = x' = x_1$,
- α is an RR-factor if $x = x' = x_2$,
- α is an LR-factor if $x = x_1$ and $x' = x_2$,
- α is an RL-factor if $x = x_2$ and $x' = x_1$.

In Fig. 2 we see that α is an LL-factor, β, δ are LR-factors, ζ is an RR-factor, and γ is an RL-factor.

Definition 4. Let ρ be a run and $I = [x_1, x_2]$ an interval of ρ. Let h_i be the length of the crossing sequence $\rho|x_i$ for both $i = 1$ and $i = 2$.

The *flow* F_I of I is a directed graph with set of nodes $\{0, \ldots, \max(h_1, h_2) - 1\}$ and set of edges consisting of all (y, y') such that there exists a factor of ρ intercepted by I that starts at location (x_i, y) and ends at location (x_j, y'), for $i, j \in \{1, 2\}$.

The *effect* E_I of I is the triple (F_I, c_1, c_2), where $c_i = \rho|x_i$ is the crossing sequence at x_i.

For example, the interval I of Fig. 2 has the flow graph $0 \mapsto 1 \mapsto 3 \mapsto 4 \mapsto 2 \mapsto 0$. It is easy to see that every node of a flow F_I has at most one incoming and at most one outgoing edge. More precisely, if $y < h_1$ is even, then it has one outgoing edge (corresponding to an LR- or LL-factor intercepted by I), and if it is odd it has one incoming edge (corresponding to an RL- or LL-factor intercepted by I). Similarly, if $y < h_2$ is even, then it has one incoming edge (corresponding to an LR- or RR-factor), and if it is odd it has one outgoing edge (corresponding to an RL- or RR-factor).

In the following we consider generic effects that are not necessarily associated with intervals of specific runs. The definition of such effects should be clear: these are triples consisting of a graph (called flow) and two crossing sequences of lengths $h_1, h_2 \leqslant h_{\mathsf{max}}$, with sets of nodes of the form $\{0, \ldots, \max(h_1, h_2) - 1\}$, that satisfy the in/out-degree properties stated above.

It is convenient to distinguish the edges in a flow based on the parity of the source and target nodes. Formally, we partition any flow F into the following subgraphs:

- F_{LR} consists of all edges of F between pairs of even nodes,
- F_{RL} consists of all edges of F between pairs of odd nodes,
- F_{LL} consists of all edges of F from an even node to an odd node,
- F_{RR} consists of all edges of F from an odd node to an even node.

We denote by $\mathcal{F}$ (resp. $\mathcal{E}$) the set of all flows (resp. effects) augmented with a dummy element $\perp$. We equip both sets $\mathcal{F}$ and $\mathcal{E}$ with a semigroup structure, where the corresponding products $\circ$ and $\odot$ are defined below (similar definitions appear in [6]). We need this semigroup structure in order to identify *idempotent loops*, that play a crucial role in our characterization of one-way definability.

Definition 5. For two graphs G, G', we denote by $G \cdot G'$ the graph with edges of the form (y, y'') such that (y, y') is an edge of G and (y', y'') is an edge of G', for some node y' that belongs to both G and G'. Similarly, we denote by G^* the graph with edges (y, y') such that there exists a (possibly empty) path in G from y to y'.

The product of two flows F, F' is the unique flow $F \circ F'$ (if it exists) such that:

- $(F \circ F')_{\mathsf{LR}} = F_{\mathsf{LR}} \cdot (F'_{\mathsf{LL}} \cdot F_{\mathsf{RR}})^* \cdot F'_{\mathsf{LR}}$,
- $(F \circ F')_{\mathsf{RL}} = F'_{\mathsf{RL}} \cdot (F_{\mathsf{RR}} \cdot F'_{\mathsf{LL}})^* \cdot F_{\mathsf{RL}}$,

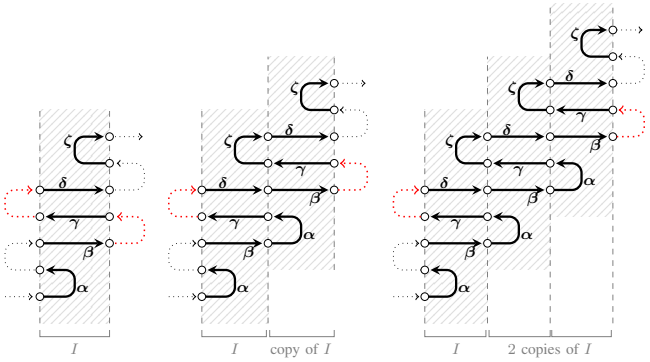


Figure 3. Pumping a loop in a run.

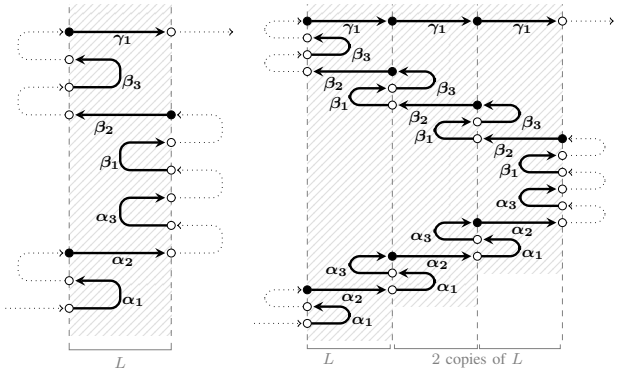


Figure 4. Pumping an idempotent loop with three components.

- $(F \circ F')_{\mathsf{LL}} = F_{\mathsf{LL}} \cup F_{\mathsf{LR}} \cdot (F'_{\mathsf{LL}} \cdot F_{\mathsf{RR}})^* \cdot F'_{\mathsf{LL}} \cdot F_{\mathsf{RL}}$,
- $(F \circ F')_{\mathsf{RR}} = F'_{\mathsf{RR}} \cup F'_{\mathsf{RL}} \cdot (F_{\mathsf{RR}} \cdot F'_{\mathsf{LL}})^* \cdot F_{\mathsf{RR}} \cdot F'_{\mathsf{LR}}$.

If no flow $F \circ F'$ exists with the above properties, then we let $F \circ F' = \bot$.

The product of two effects $E = (F, c_1, c_2)$ and $E' = (F', c'_1, c'_2)$ is either the effect $E \odot E' = (F \circ F', c_1, c'_2)$ or the dummy element $\bot$, depending on whether $F \circ F' \neq \bot$ and $c_2 = c'_1$.

For example, let F be the flow of interval I in Fig. 2. Then $(F \circ F)_{\mathsf{LL}} = \{(0,1),(2,3)\}$, $(F \circ F)_{\mathsf{RR}} = \{(1,2),(3,4)\}$, and $(F \circ F)_{\mathsf{LR}} = \{(4,0)\}$ — one can quickly verify this with the help of Fig. 3.

It is also easy to see that $(\mathcal{F}, \circ)$ and $(\mathcal{E}, \odot)$ are finite semigroups, and that for every run ρ and every pair of consecutive intervals $I = [x_1, x_2]$ and $J = [x_2, x_3]$ of ρ, $F_{I \cup J} = F_I \circ F_J$ and $E_{I \cup J} = E_I \odot E_J$. In particular, the function E that associates each interval I of ρ with the corresponding effect E_I can be seen as a semigroup homomorphism.

Note that, in a normalized successful run, there are at most $|Q|^{h_{\max}}$ distinct crossing sequences and at most $4^{h_{\max}}$ distinct flows, since there are at most $h_{\max}$ edges in a flow, and each one has one of the 4 possible types $\mathsf{LL}, \ldots, \mathsf{RR}$. Hence there are at most $(2|Q|)^{2h_{\max}}$ distinct effects.

Loops and components. Loops of a two-way run are the basic building blocks for characterizing one-way definability. We will consider special types of loops, called idempotent loops, when showing that outputs generated in non left-to-right manner are essentially periodic.

Definition 6. A *loop* of ρ is an interval $L = [x_1, x_2]$ whose endpoints have the same crossing sequences, i.e. $\rho|x_1 = \rho|x_2$. It is said to be *idempotent* if $E_L = E_L \odot E_L$ and $E_L \neq \bot$.

For example, the interval I of Fig. 2 is a loop, if one assumes that the crossing sequences at the borders of I are the same. However, by comparing with Fig. 3, it is easy to see that I is not idempotent. On the other hand, the loop consisting of 2 copies of I is idempotent.

Given a loop $L = [x_1, x_2]$ and a number $m \in \mathbb{N}$, we can introduce m new copies of L and connect the intercepted factors in the obvious way. Fig. 3 shows how to do this for $m = 1$ and $m = 2$. The operation that we just described is called *pumping*, and results in a new run of the transducer $\mathcal{T}$ on the word

$$\mathsf{pump}_L^{m+1}(u) := u[0, x_1] \cdot \big(u[x_1{+}1, x_2]\big)^{m+1} \cdot u[x_2{+}1, \omega] \,.$$

We denote by $\mathsf{pump}_L^{m+1}(\rho)$ the pumped[1] run on $\mathsf{pump}_L^{m+1}(u)$.

The goal in this section is to describe the shape of the pumped run $\mathsf{pump}_L^{m+1}(\rho)$ (and the produced output as well) when L is an *idempotent* loop. We will focus on idempotent loops because pumping non-idempotent loops may induce permutations of outputs that are difficult to handle. For example, if we consider again the non-idempotent loop I to the left of Fig. 3, the factor of the run between β and γ (to the right of I, highlighted in red) precedes the factor between γ and δ (to the left of I, again in red), but this ordering is reversed when a new copy of I is added.

When pumping a loop L, subsets of factors intercepted by L are glued together to form longer factors intercepted by the unioned copies of L. The concept of component that we introduce below aims at identifying the groups of factors that are glued together.

Definition 7. A *component* of a loop L is any strongly connected component of its flow F_L (note that this is also a cycle, since every node in it has in/out-degree 1). Given a component C, we denote by $\min(C)$ (resp. $\max(C)$) the minimum (resp. maximum) node in C. We say that C is *left-to-right* (resp. *right-to-left*) if $\min(C)$ is even (resp., odd). An (L, C)*-factor* is a factor of the run that is intercepted by L and corresponds to an edge of C.

For example, the loop I of Fig. 3 contains a single component $C = \{0 \mapsto 1 \mapsto 3 \mapsto 4 \mapsto 2 \mapsto 0\}$ which is left-to-right. Another example is given in Fig. 4, where the loop L has three components C_1, C_2, C_3 (ordered from bottom to top): $\alpha_1, \alpha_2, \alpha_3$ are the (L, C_1)-factors, $\beta_1, \beta_2, \beta_3$ are the (L, C_2)-factors, and γ_1 is the unique (L, C_3)-factor.

We will usually list the (L, C)-factors based on their order of occurrence in the run.

1. Using similar constructions, one could remove a loop L from a run ρ, resulting in the run $\mathsf{pump}_L^0(\rho)$. As we do not need this, the operation pump_L will always be parametrized by a positive number $m + 1$.

The following lemma describes the precise shape and order of such factors when the loop L is idempotent. It can be used to reason on the shape of runs obtained by pumping idempotent loops.

Lemma 8. If C is a left-to-right (resp. right-to-left) component of an *idempotent* loop L, then the (L, C)-factors are in the following order: k LL-factors (resp. RR-factors), followed by one LR-factor (resp. RL-factor), followed by k RR-factors (resp. LL-factors), for some $k \geqslant 0$.

We also need to introduce the notions of *anchor* (Def. 9) and *trace* (Def. 10).

Definition 9. Let C be a component of an idempotent loop $L = [x_1, x_2]$. The *anchor* of C inside L, denoted[2] $\mathsf{an}(C)$, is either the location $\big(x_1, \max(C)\big)$ or the location $\big(x_2, \max(C)\big)$, depending on whether C is left-to-right or right-to-left.

Intuitively, the anchor $\mathsf{an}(C)$ of a component C of L is the source location of the unique LR- or RL-factor intercepted by L that corresponds to an edge of C (recall Lemma 8). In Fig. 4 the anchors are represented by the black dots.

Definition 10. Let C be a component of some idempotent loop L and let $(i_0, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k), (i_k, i_{k+1})$ be a cycle of C, where $i_0 = i_{k+1} = \max(C)$. For every $j = 0, \ldots, k$, let β_j be the factor intercepted by L that corresponds to the edge (i_j, i_{j+1}) of C. The *trace* of C inside L is the run $\mathsf{tr}(C) = \beta_0\ \beta_1\ \cdots\ \beta_k$ (note that this is not necessarily a factor of the original run ρ).

Intuitively, the trace $\mathsf{tr}(C)$ is obtained by concatenating the (L, C)-factors together, where the first factor is the (unique) LR-/RL-factor that starts at the anchor $\mathsf{an}(C)$ and the remaining ones are the LL-factors interleaved with the RR-factors.

For example, by referring again to the components C_1, C_2, C_3 of Fig. 4, we have the following traces: $\mathsf{tr}(C_1) = \alpha_2\ \alpha_1\ \alpha_3$, $\mathsf{tr}(C_2) = \beta_2\ \beta_1\ \beta_3$, and $\mathsf{tr}(C_3) = \gamma_1$.

As shown by the following proposition, iterations of idempotent loops translate to iterations of traces $\mathsf{tr}(C)$ of components.

Proposition 11. Let L be an idempotent loop of ρ with components $C_1, \ldots, C_k$, listed according to the order of their anchors: $\mathsf{an}(C_1) \lhd \cdots \lhd \mathsf{an}(C_k)$. For all $m \in \mathbb{N}$, we have

$$\mathsf{pump}_L^{m+1}(\rho) \;=\; \rho_0\ \mathsf{tr}(C_1)^m\ \rho_1\ \cdots\ \rho_{k-1}\ \mathsf{tr}(C_k)^m\ \rho_k$$

where

- ρ_0 is the prefix of ρ that ends at $\mathsf{an}(C_1)$,
- ρ_i is the factor $\rho[\mathsf{an}(C_i), \mathsf{an}(C_{i+1})]$, for all $1 \leqslant i < k$,
- ρ_k is the suffix of ρ that starts at $\mathsf{an}(C_k)$.

For example, referring to the left hand-side of Fig. 4, the run ρ_0 goes until the first location marked by a black dot. The runs ρ_1 and ρ_2, resp., are between the first and the second black dot, and the second and third black dot. Finally, ρ_3 is the suffix starting at the last black dot. The pumped run $\mathsf{pump}_L^{m+1}(\rho)$ for $m = 2$ is depicted to the right of Fig. 4.

Ramsey-type arguments. We conclude the section by describing a technique that can be used for bounding the length of the outputs produced by factors of the run ρ. This technique is based on Ramsey-type arguments and relies on Simon's "factorization forest" theorem [9], [24], which we recall below.

Let X be a set of positions of ρ. A *factorization forest* for X is an unranked tree, where the nodes are intervals I with endpoints in X, labelled with the corresponding effect E_I, the ancestor relation is given by the containment order on intervals, the leaves are the minimal intervals $[x_1, x_2]$, with x_2 successor of x_1 in X, and for every internal node I with children $J_1, \ldots, J_k$, we have:

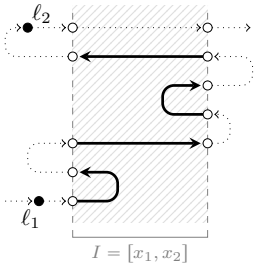
- $I = J_1 \cup \cdots \cup J_k$,
- $E_I = E_{J_1} \odot \cdots \odot E_{J_k}$,
- if $k > 2$, then $E_I = E_{J_1} = \cdots = E_{J_k}$ is an idempotent of the semigroup $(\mathcal{E}, \odot)$.

We will make use of the following three constants defined from the transducer $\mathcal{T}$: the maximum number c_{max} of letters output by a single transition, the maximal length $h_{\mathsf{max}} = 2|Q| - 1$ of a crossing sequence, and the maximal size $e_{\mathsf{max}} = (2|Q|)^{2h_{\mathsf{max}}}$ of the effect semigroup $(\mathcal{E}, \odot)$. By $\boldsymbol{B} = \boldsymbol{c_{\mathsf{max}}} \cdot \boldsymbol{h_{\mathsf{max}}} \cdot (\mathbf{2}^{\mathbf{3}\boldsymbol{e_{\mathsf{max}}}} + \mathbf{4})$ we will denote the main constant appearing in all subsequent sections.

Theorem 12 (Factorization forest theorem [9], [24])**.** For every set X of positions of ρ, there is a factorization forest for X of height at most $3e_{\mathsf{max}}$.

It is easy to use the above theorem to show that every run that produces an output longer than $\boldsymbol{B}$ contains an idempotent loop with non-empty output. Below, we present a result in the same spirit, but refined in a way that it can be used to find anchors of components of loops inside specific intervals.

In order to state it formally, we need to consider subsequences of ρ induced by sets of locations that are not necessarily intervals. Recall that $\rho[\ell_1, \ell_2]$ denotes the factor of ρ delimited by two locations $\ell_1 \trianglelefteq \ell_2$. Similarly, given any set Z of (possibly non-consecutive) locations, we denote by $\rho \mid Z$ the subsequence of ρ induced by Z. A transition of $\rho \mid Z$ is a transition from some ℓ to ℓ', where both ℓ, ℓ' belong to Z. The output $\mathsf{out}(\rho \mid Z)$ is the concatenation of the outputs of the transitions of $\rho \mid Z$ (in the order given by ρ). An example of subrun $\rho \mid Z$ is represented by the thick arrows in the figure to the right, where $Z = [\ell_1, \ell_2] \cap (I \times \mathbb{N})$.



Theorem 13. Let $I = [x_1, x_2]$ be an interval of positions, $K = [\ell_1, \ell_2]$ an interval of locations, and $Z = K \cap (I \times \mathbb{N})$.

2. In denoting the anchor — and similarly the trace — of a component C inside a loop L, we omit the annotation specifying L, since this is often understood from the context.

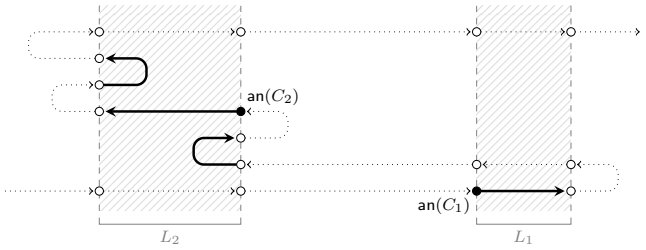


Figure 5. An inversion with components intercepting the highlighted factors.

If $\big|\mathsf{out}(\rho \mid Z)\big| > \boldsymbol{B}$, then there exist an idempotent loop L and a component C of L such that

- $x_1 < \min(L) < \max(L) < x_2$ (in particular, $L \subsetneq I$),
- $\ell_1 \lhd \mathsf{an}(C) \lhd \ell_2$ (in particular, $\mathsf{an}(C) \in K$),
- $\mathsf{out}(\mathsf{tr}(C)) \neq \varepsilon$.

5. Inversions and periods

As suggested by Examples 1 and 2, a typical phenomenon that may prevent a transducer from being one-way definable is that of a non-periodic *inversion*. An inversion essentially corresponds to a long output produced from right to left. The main result in this section is Proposition 16, that shows that for a one-way definable transduction, the output between the locations delimiting an inversion must have bounded period.

Definition 14. An *inversion* of ρ is a tuple (L_1, C_1, L_2, C_2) such that

- L_i is an idempotent loop, for both $i = 1, 2$,
- C_i is a component of L_i, for both $i = 1, 2$,
- $\mathsf{an}(C_1) \lhd \mathsf{an}(C_2)$,
- $\mathsf{an}(C_i) = (x_i, y_i)$, for both $i = 1, 2$, and $x_1 \geqslant x_2$,
- both $\mathsf{out}(\mathsf{tr}(C_1))$ and $\mathsf{out}(\mathsf{tr}(C_2))$ are non-empty.

Fig. 5 gives an example of an inversion involving the loop L_1 with its first component and the loop L_2 with its second component (we highlighted the anchors and the factors corresponding to these components).

Definition 15. A word $w = a_1 \cdots a_n$ has *period* p if $a_i = a_{i+p}$ for all pairs of positions $i, i+p$ of w.

For example, $w = abc\,abc\,ab$ has period 3.

One-way definability of functional two-way transducers essentially amounts to showing that the output produced by every inversion has bounded period. The proposition below shows a slightly stronger periodicity property, which refers to the output produced inside the inversion extended on both sides by the trace outputs. We will need this stronger property later, when dealing with overlapping portions of the run delimited by different inversions.

Proposition 16. If $\mathcal{T}$ is one-way definable, then for every inversion (L_1, C_1, L_2, C_2) of a successful run ρ of $\mathcal{T}$, the word

$$\mathsf{out}\big(\mathsf{tr}(C_1)\big)\ \mathsf{out}\big(\rho[\mathsf{an}(C_1), \mathsf{an}(C_2)]\big)\ \mathsf{out}\big(\mathsf{tr}(C_2)\big)$$

has period p that divides both $|\mathsf{out}(\mathsf{tr}(C_1))|$ and $|\mathsf{out}(\mathsf{tr}(C_2))|$. Moreover, $p \leqslant \boldsymbol{B}$.

The basic combinatorial argument for proving Proposition 16 is a classical result in word combinatorics called Fine and Wilf's theorem [16]. Essentially, the theorem says that, whenever two periodic words w_1, w_2 share a sufficiently long factor, then they have as period the greatest common divisor of the two original periods. Below, we state a slightly stronger variant of Fine-Wilf's theorem, which contains an additional claim showing how to align a common factor of the words w_1, w_2 so as to form a third word w_3 that contains a prefix of w_1 and a suffix of w_2. The additional claim will be fully exploited in the proof of Proposition 26.

Lemma 17 (Fine-Wilf's theorem)**.** If $w_1 = w_1' \, w \, w_1''$ has period p_1, $w_2 = w_2' \, w \, w_2''$ has period p_2, and the common factor w has length at least $p_1 + p_2 - \gcd(p_1, p_2)$, then w_1, w_2, and $w_3 = w_1' \, w \, w_2''$ have period $\gcd(p_1, p_2)$.

Two further combinatorial results are central in the proof of Proposition 16. The first one is a result of Kortelainen [19], later improved and simplified by Saarela [21]. It is related to word equations with iterated factors, like those that arise from considering outputs of pumped versions of a run. To improve readability, we highlight the important iterations of factors inside the considered equations.

Theorem 18 (Theorem 4.3 in [21])**.** Consider a word equation

$$v_0 \boldsymbol{v_1^m} v_2 \ldots v_{k-1} \boldsymbol{v_k^m} v_{k+1} \;=\; w_0 \boldsymbol{w_1^m} w_2 \ldots w_{k'-1} \boldsymbol{w_{k'}^m} w_{k'+1}$$

where m is the unknown and v_i, w_j are words. Then the set of solutions of the equation is either finite or $\mathbb{N}$.

The second combinatorial result considers a word equation with iterated factors parametrized by two unknowns m_1, m_2 that occur in opposite order in the left, respectively right hand-side of the equation. This type of equation arises when we compare the output associated with an inversion of $\mathcal{T}$ and the output produced by an equivalent one-way transducer $\mathcal{T}'$.

Lemma 19. Consider a word equation of the form

$$v_0^{(m_1,m_2)} \boldsymbol{v_1^{m_1}} v_2^{(m_1,m_2)} \boldsymbol{v_3^{m_2}} v_4^{(m_1,m_2)} \;=\; w_0 \boldsymbol{w_1^{m_2}} w_2 \boldsymbol{w_3^{m_1}} w_4$$

where m_1, m_2 are the unknowns, v_1, v_3 are non-empty words, and $v_0^{(m_1,m_2)}, v_2^{(m_1,m_2)}, v_4^{(m_1,m_2)}$ are words that may contain factors of the form v^{m_1} or v^{m_2} (for some word v). If the above equation holds for all $m_1, m_2 \in \mathbb{N}$, then the words $\boldsymbol{v_1}\ \boldsymbol{v_1^{m_1}}\ v_2^{(m_1,m_2)}\ \boldsymbol{v_3^{m_2}}\ \boldsymbol{v_3}$ are periodic with period $\gcd(|v_1|, |v_3|)$, for all $m_1, m_2 \in \mathbb{N}$.

The last ingredient used in the proof of Proposition 16 is a bound on the period of the output produced by an inversion. For this, we introduce a suitable notion of minimality of loops and loop components:

Definition 20. Consider pairs (L, C) consisting of an idempotent loop L and a component C of L.

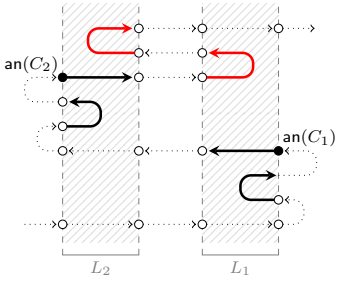
- On such pairs, we define the relation $\sqsubset$ by $(L', C') \sqsubset (L, C)$ if $L' \subsetneq L$ and at least one (L', C')-factor is contained in some (L, C)-factor.
- A pair (L, C) is *output-minimal* if for all pairs $(L', C') \sqsubset (L, C)$, we have $\mathsf{out}(\mathsf{tr}(C')) = \varepsilon$.

Note that the relation $\sqsubset$ is not a partial order in general (it is however antisymmetric). Lemma 21 below shows that the length of the output trace of C inside L is bounded whenever (L, C) is output-minimal.

Lemma 21. For every output-minimal pair (L, C), $|\mathsf{out}(\mathsf{tr}(C))| \leqslant \boldsymbol{B}$.

Proof sketch. We use a Ramsey-type argument here: if $|\mathsf{out}(\mathsf{tr}(C))| > \boldsymbol{B}$, then Theorem 13 can be applied to exhibit an idempotent loop strictly inside L and a component C of it with non-empty trace output. This would contradict the output-minimality of (L, C). □

We remark that the above lemma cannot be used directly to bound the period of the output produced by an inversion. The reason is that we cannot assume that inversions are built up from output-minimal pairs. An example is given in the figure to the right, which shows a run where the only inversion (L_1, C_1, L_2, C_2) contains pairs that are not output-minimal: the factors that produce long outputs are those in red, but they occur outside $\rho[\mathsf{an}(C_1), \mathsf{an}(C_2)]$.



We are now ready to sketch the proof of Proposition 16.

Proof sketch of Proposition 16. In the first half of the proof we pump the two loops L_1 and L_2 so that we obtain also loops in the assumed equivalent one-way transducer $\mathcal{T}'$. We then consider the outputs of the pumped runs of $\mathcal{T}$ and $\mathcal{T}'$, which contain iterated factors parametrized by two natural numbers m_1, m_2. As those outputs must agree due to the equivalence of $\mathcal{T}$ and $\mathcal{T}'$, we get an equation as in Lemma 19, where the word v_1 belongs to $\mathsf{out}(\mathsf{tr}(C_1))^+$ and the word v_3 belongs to $\mathsf{out}(\mathsf{tr}(C_2))^+$. Lemma 19 shows that the word described by the equation has period p dividing $\gcd(|v_1|, |v_3|)$, and Lemma 17 shows that p even divides $|\mathsf{out}(\mathsf{tr}(C_1))|$ and $|\mathsf{out}(\mathsf{tr}(C_2))|$. Finally, we use Theorem 18 to transfer the periodicity property from the word of the equation to the word $w = \mathsf{out}(\mathsf{tr}(C_1))\ \mathsf{out}(\rho[\mathsf{an}(C_1), \mathsf{an}(C_2)])\ \mathsf{out}(\mathsf{tr}(C_2))$ produced by the original run of $\mathcal{T}$. This is possible because the word of the equation is obtained by iterating factors of w. In particular, by reasoning separately on the parameters that define those iterations, and by stating the periodicity property as an equation in the form required by Theorem 18, we show that the periodicity equation holds on all parameters, and thus in particular on w.

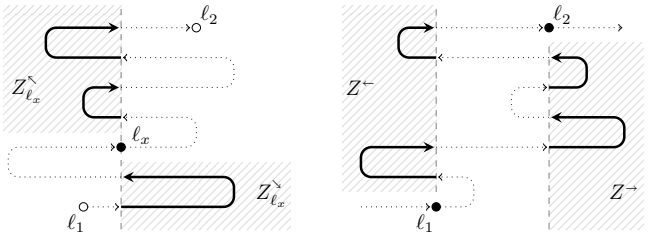


Figure 6. Outputs that need to be bounded in a diagonal and in a block.

In the second half of the proof we show that the period p is bounded by $\boldsymbol{B}$. This requires a refinement of the previous arguments and involves pumping the run of $\mathcal{T}$ simultaneously on three different loops. The idea is that by pumping we manage to find inversions with some output-minimal pair (L_0, C_0). In this way we show that the period p also divides $\mathsf{out}(\mathsf{tr}(C_0))$, which is bounded by $\boldsymbol{B}$ according to Lemma 21. □

6. One-way definability

Proposition 16 is the main combinatorial argument for characterizing two-way transducers that are one-way definable. In this section we provide the remaining arguments. Roughly, the idea is to decompose every successful run ρ into factors that produce long outputs either in a left-to-right manner ("diagonals"), or based on an almost periodic pattern ("blocks").

We say that a word w is *almost periodic with bound* p if $w = w_0\ w_1\ w_2$ for some words w_0, w_2 of length at most p and some word w_1 of period at most p.

We illustrate the following definition in Fig. 6.

Definition 22. Consider a factor $\rho[\ell_1, \ell_2]$ of the run, where $\ell_1 = (x_1, y_1)$, $\ell_2 = (x_2, y_2)$, and $x_1 \leqslant x_2$. We call $\rho[\ell_1, \ell_2]$

- a *diagonal* if for all $x \in [x_1, x_2]$, there is a location ℓ_x at position x such that $\ell_1 \trianglelefteq \ell_x \trianglelefteq \ell_2$ and the words $\mathsf{out}(\rho \mid Z^{\nwarrow}_{\ell_x})$ and $\mathsf{out}(\rho \mid Z^{\searrow}_{\ell_x})$ have length at most $\boldsymbol{B}$, where $Z^{\nwarrow}_{\ell_x} = [\ell_x, \ell_2] \cap \big([0, x] \times \mathbb{N}\big)$ and $Z^{\searrow}_{\ell_x} = [\ell_1, \ell_x] \cap \big([x, \omega] \times \mathbb{N}\big)$;
- a *block* if the word $\mathsf{out}(\rho[\ell_1, \ell_2])$ is almost periodic with bound $\boldsymbol{B}$, and $\mathsf{out}(\rho \mid Z^{\leftarrow})$ and $\mathsf{out}(\rho \mid Z^{\rightarrow})$ have length at most $\boldsymbol{B}$, where $Z^{\leftarrow} = [\ell_1, \ell_2] \cap \big([0, x_1] \times \mathbb{N}\big)$ and $Z^{\rightarrow} = [\ell_1, \ell_2] \cap \big([x_2, \omega] \times \mathbb{N}\big)$.

Intuitively, the output of a diagonal $\rho[\ell_1, \ell_2]$ can be simulated while scanning the input interval $[x_1, x_2]$ from left to right, since the outputs of $\rho \mid Z^{\nwarrow}_{\ell_x}$ and $\rho \mid Z^{\searrow}_{\ell_x}$ are bounded. A similar argument applies to a block $\rho[\ell_1, \ell_2]$, where in addition, one exploits the fact that the output is almost periodic. Roughly, the idea is that one can simulate the output of a block by outputting symbols according to a periodic pattern, and in a number that is determined from the transitions on $u[x_1, x_2]$ and the guessed (bounded) outputs on $Z^{\leftarrow}$ and $Z^{\rightarrow}$.

The general idea for turning a two-way transducer $\mathcal{T}$ into an equivalent one-way transducer $\mathcal{T}'$ is to guess (and

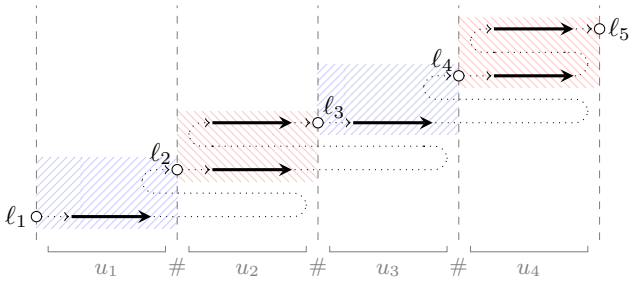


Figure 7. A decomposition of a run of a two-way transducer.

check) a factorization of a successful run of $\mathcal{T}$ into factors that are either diagonals or blocks, and properly arranged following the order of positions.

Definition 23. A *decomposition* of ρ is a factorization $\prod_i \rho[\ell_i, \ell_{i+1}]$ of ρ into diagonals and blocks, where $\ell_i = (x_i, y_i)$ and $x_i < x_{i+1}$ for all i.

The one-way transducer $\mathcal{T}'$ whose existence is stated by Theorem 3 simulates $\mathcal{T}$ precisely on those inputs u that have some successful run admitting a decomposition. To provide further intuition on the notion of decomposition, we consider again the transduction of Example 2 and the two-way transducer $\mathcal{T}$ that implements it in the most natural way. Fig. 7 shows an example of a run of $\mathcal{T}$ on an input of the form $u_1 \# u_2 \# u_3 \# u_4$, where $u_2, u_4 \in (abc)^*$, $u_1\, u_3 \notin (abc)^*$, and u_3 has even length. The factors of the run that produce long outputs are highlighted by the bold arrows. The first and third factors of the decomposition, i.e. $\rho[\ell_1, \ell_2]$ and $\rho[\ell_3, \ell_4]$, are diagonals (represented by the blue/rightward hatched areas); the second and fourth factors $\rho[\ell_2, \ell_3]$ and $\rho[\ell_4, \ell_5]$ are blocks (represented by the red/leftward hatched areas).

Theorem 24. Let $\mathcal{T}$ be a functional two-way transducer. The following are equivalent:

P1) $\mathcal{T}$ is one-way definable.

P2) For all inversions (L_1, C_1, L_2, C_2) of all successful runs of $\mathcal{T}$, the word

$$\mathsf{out}\big(\mathsf{tr}(C_1)\big)\ \mathsf{out}\big(\rho[\mathsf{an}(C_1), \mathsf{an}(C_2)]\big)\ \mathsf{out}\big(\mathsf{tr}(C_2)\big)$$

has period $p \leqslant \boldsymbol{B}$ dividing $|\mathsf{out}(\mathsf{tr}(C_1))|$, $|\mathsf{out}(\mathsf{tr}(C_2))|$.

P3) Every successful run of $\mathcal{T}$ admits a decomposition.

The implication from **P1** to **P2** was already shown in Proposition 16. The rest of this section is devoted to prove the implications from **P2** to **P3** and from **P3** to **P1**. The issues related to the complexity of the characterization will be discussed further below.

From periodicity to existence of decompositions (P2→P3). As usual, we fix a successful run ρ of $\mathcal{T}$. We will prove a slightly stronger result than the implication from **P2** to **P3**, namely: if every inversion of ρ satisfies the periodicity property stated in **P2**, then ρ admits a decomposition (note that this is independent of whether other runs satisfy or not **P2**). To identify the blocks of a possible decomposition of ρ we consider a suitable equivalence relation between locations:

Definition 25. A location ℓ is *covered* by an inversion (L_1, C_1, L_2, C_2) if $\mathsf{an}(C_1) \trianglelefteq \ell \trianglelefteq \mathsf{an}(C_2)$. We define the relation S by letting $\ell\ \mathsf{S}\ \ell'$ if ℓ, ℓ' are covered by the *same* inversion. We define the equivalence relation S^* as the reflexive and transitive closure of S.

Locations covered by the same inversion (L_1, C_1, L_2, C_2) yield an interval w.r.t. the run ordering $\trianglelefteq$. Thus every non-singleton S^*-class can be seen as a union of such intervals, say $K_1, \ldots, K_m$, that are two-by-two overlapping, namely, $K_i \cap K_{i+1} \neq \varnothing$ for all $i < m$. In particular, a non-singleton S^*-class is an interval of locations witnessed by a series of inversions $(L_{2i}, C_{2i}, L_{2i+1}, C_{2i+1})$ such that $\mathsf{an}(C_{2i}) \trianglelefteq \mathsf{an}(C_{2i+2}) \trianglelefteq \mathsf{an}(C_{2i+1}) \trianglelefteq \mathsf{an}(C_{2i+3})$.

The next result exploits the shape of a non-singleton S^*-class, the assumption that ρ satisfies the periodicity property stated in **P2**, and Lemma 17, to show that the output produced inside an S^*-class has bounded period.

Proposition 26. If ρ satisfies the periodicity property stated in **P2** and $\ell \trianglelefteq \ell'$ are two locations in the same S^*-class, then $\mathsf{out}\big(\rho[\ell, \ell']\big)$ has period at most $\boldsymbol{B}$.

The S^*-classes considered so far cannot be directly used as blocks for the desired decomposition of ρ, since the x-coordinates of their endpoints might not be in the appropriate order. The next definition takes care of this, by enlarging the S^*-classes according to x-coordinates of anchors.

Definition 27. Let $K = [\ell, \ell']$ be a non-singleton S^*-class, let $\mathsf{an}(K)$ be the restriction of K to the locations that are anchors of components of inversions, and let $X_{\mathsf{an}(K)} = \{x : \exists y\, (x, y) \in \mathsf{an}(K)\}$ be the projection of $\mathsf{an}(K)$ on positions. We define $\mathsf{block}(K) = [\ell_1, \ell_2]$, where

- ℓ_1 is the latest location $(x, y) \trianglelefteq \ell$ such that $x = \min\big(X_{\mathsf{an}(K)}\big)$,
- ℓ_2 is the earliest location $(x, y) \trianglerighteq \ell'$ such that $x = \max\big(X_{\mathsf{an}(K)}\big)$

(note that the location ℓ_1 exists since ℓ is the anchor of the first component of an inversion, and ℓ_2 exists for similar reasons).

Lemma 28. If $K = [\ell, \ell']$ is a non-singleton S^*-class, then $\rho[\ell_1, \ell_2]$ is a block, where $[\ell_1, \ell_2] = \mathsf{block}(K)$.

Proof sketch. The periodicity of $\mathsf{out}(\rho[\ell, \ell'])$ is obtained by applying Proposition 26. Then Theorem 13 is applied twice: first to bound $\mathsf{out}(\rho[\ell_1, \ell])$ and $\mathsf{out}(\rho[\ell', \ell_2])$ (hence proving that $\mathsf{out}(\rho[\ell_1, \ell_2])$ is almost periodic with bound $\boldsymbol{B}$), and second, to bound $\mathsf{out}(\rho \mid Z^{\leftarrow})$ and $\mathsf{out}(\rho \mid Z^{\rightarrow})$, as introduced in Definition 22. □

The next lemma shows that blocks do not overlap along the input axis:

Lemma 29. Suppose that K_1 and K_2 are two different non-singleton S^*-classes such that $\ell \lhd \ell'$ for all $\ell \in K_1$ and

$\ell' \in K_2$. Let $\mathsf{block}(K_1) = [\ell_1, \ell_2]$ and $\mathsf{block}(K_2) = [\ell_3, \ell_4]$, with $\ell_2 = (x_2, y_2)$ and $\ell_3 = (x_3, y_3)$. Then $x_2 < x_3$.

For the sake of brevity, we call S^*-*block* any factor of the form $\rho \mid \mathsf{block}(K)$ that is obtained by applying Definition 27 to a non-singleton S^*-class K. The results obtained so far imply that every location covered by an inversion is also covered by an S^*-block (Lemma 28), and that the order of occurrence of S^*-blocks is the same as the order of positions (Lemma 29). So the S^*-blocks can be used as factors for the decomposition of ρ we are looking for. Below, we show that the remaining factors of ρ, which do not overlap the S^*-blocks, are diagonals. This will complete the construction of a decomposition of ρ.

Formally, we say that a factor $\rho[\ell_1, \ell_2]$ *overlaps* another factor $\rho[\ell_3, \ell_4]$ if $[\ell_1, \ell_2] \cap [\ell_3, \ell_4] \neq \varnothing$, $\ell_2 \neq \ell_3$, and $\ell_1 \neq \ell_4$.

Lemma 30. Let $\rho[\ell_1, \ell_2]$ be a factor of ρ that does not overlap any S^*-block, with $\ell_1 = (x_1, y_1)$, $\ell_2 = (x_2, y_2)$, and $x_1 < x_2$. Then $\rho[\ell_1, \ell_2]$ is a diagonal.

Proof sketch. If $\rho[\ell_1, \ell_2]$ is not a diagonal, we can find a location $\ell_1 \trianglelefteq \ell \trianglelefteq \ell_2$ for which $|\mathsf{out}(\rho \mid Z_\ell^{\nwarrow})| > \boldsymbol{B}$ and $|\mathsf{out}(\rho \mid Z_\ell^{\searrow})| > \boldsymbol{B}$ (recall Definition 22). By applying again Theorem 13, we derive the existence of an inversion between ℓ_1 and ℓ_2, and thus of an S^*-block overlapping $\rho[\ell_1, \ell_2]$. □

From decompositions to one-way definability (P3→P1). Hereafter, we denote by U the language of words $u \in \mathsf{dom}(\mathcal{T})$ such that *all* successful runs of $\mathcal{T}$ on u admit a decomposition.

So far, we know that if $\mathcal{T}$ is one-way definable (**P1**), then $U = \mathsf{dom}(\mathcal{T})$ (**P3**). This reduces the one-way definability problem for $\mathcal{T}$ to the containment problem $\mathsf{dom}(\mathcal{T}) \subseteq U$. We will see later how the latter problem can be decided in double exponential space by further reducing it to checking the emptiness of the intersection of the languages $\mathsf{dom}(\mathcal{T})$ and $U^{\complement}$, where $U^{\complement}$ is the complement of U.

The next proposition says that a one-way transducer $\mathcal{T}'$ of triple exponential size can be constructed such that

$$\mathcal{T}' \subseteq \mathcal{T} \qquad \text{and} \qquad \mathsf{dom}(\mathcal{T}') \supseteq U.$$

In particular, the existence of such a transducer $\mathcal{T}'$ proves the implication from **P3** to **P1** of Theorem 24. It also proves the second item of Theorem 3, because when $\mathcal{T}$ is one-way definable, $U = \mathsf{dom}(\mathcal{T})$, and hence $\mathcal{T}$ and $\mathcal{T}'$ are equivalent.

Intuitively, given an input u, the one-way transducer $\mathcal{T}'$ guesses a successful run ρ of $\mathcal{T}$ on u and a decomposition of ρ, and then use the decomposition to simulate the output produced by ρ. Note that $\mathcal{T}'$ accepts at least all the words of U, possibly more. As a matter of fact, it would be difficult to construct a transducer whose domain coincides with U, since checking membership in U involves a universal quantification.

Proposition 31. Given a functional two-way transducer $\mathcal{T}$, one can construct in 3EXPTIME a one-way transducer $\mathcal{T}'$ such that $\mathcal{T}' \subseteq \mathcal{T}$ and $\mathsf{dom}(\mathcal{T}') \supseteq U$.

Deciding one-way definability. Recall that $\mathcal{T}$ is one-way definable iff $\mathsf{dom}(\mathcal{T}) \subseteq U$, so iff $\mathsf{dom}(\mathcal{T}) \cap U^{\complement} = \varnothing$. The lemma below exploits the characterization of Theorem 24 to show that the language $U^{\complement}$ can be recognized by an NFA $\mathcal{U}^{\complement}$ of triple exponential size. The lemma actually shows that the NFA recognizing $U^{\complement}$ can be constructed using double exponential *workspace*.

Lemma 32. Given a functional two-way transducer $\mathcal{T}$, one can construct in 2EXPSPACE an NFA recognizing $U^{\complement}$.

Proof. Consider an input word u. By Theorem 24 we know that $u \in U^{\complement}$ iff there exist a successful run ρ of $\mathcal{T}$ on u and an inversion $\mathcal{I} = (L_1, C_1, L_2, C_2)$ of ρ such that no positive number $p \leqslant \boldsymbol{B}$ is a period of the word

$$w_{\rho,\mathcal{I}} = \mathsf{out}\big(\mathsf{tr}(C_1)\big)\ \mathsf{out}\big(\rho[\mathsf{an}(C_1), \mathsf{an}(C_2)]\big)\ \mathsf{out}\big(\mathsf{tr}(C_2)\big).$$

The latter condition on $w_{\rho,\mathcal{I}}$ can be rephrased as follows: there is a function $f : \{1, \ldots, \boldsymbol{B}\} \to \{1, \ldots, |w_{\rho,\mathcal{I}}|\}$ such that $w_{\rho,\mathcal{I}}[f(p)] \neq w_{\rho,\mathcal{I}}[f(p) + p]$ for all positive numbers $p \leqslant \boldsymbol{B}$. Recall that $\boldsymbol{B} = c_{\mathsf{max}} \cdot h_{\mathsf{max}} \cdot (2^{3e_{\mathsf{max}}} + 4)$, where $h_{\mathsf{max}} = 2|Q| - 1$, $e_{\mathsf{max}} = (2|Q|)^{2h_{\mathsf{max}}}$, and Q is the state space of the two-way transducer $\mathcal{T}$. This means that the run ρ, the inversion $\mathcal{I}$, and the function f described above can all be guessed within double exponential space, namely, using a number of states that is at most a triple exponential w.r.t. $|\mathcal{T}|$. In particular, we can construct in 2EXPSPACE an NFA recognizing $U^{\complement}$. □

As a consequence of the previous lemma and of Theorem 24, we have that the emptiness of the language $\mathsf{dom}(\mathcal{T}) \cap U^{\complement}$, and hence the one-way definability of $\mathcal{T}$, can be decided in 2EXPSPACE:

Corollary 33. The problem of deciding whether a functional two-way transducer is one-way definable is in 2EXPSPACE.

7. Definability by sweeping transducers

A two-way transducer is called *sweeping* if every successful run of it performs reversals only at the extremities of the input word, i.e. when reading the symbols $\vdash$ or $\dashv$. Similarly, we call it k-*pass sweeping* if it is sweeping and every successful run performs at most $k - 1$ reversals. Clearly, a 1-pass sweeping transducer is the same as a one-way transducer.

In this section we are considering the following question: given a functional two-way transducer, is it equivalent to some k-pass sweeping transducer? We call such transducers k-*pass sweeping definable*. If the parameter k is not given *a priori*, then we denote them as *sweeping definable transducers*.

In [4] we built up on the characterization of one-way definability for (the restricted class of) sweeping transducers [3] in order to determine the minimal number of passes required by sweeping transductions. Essentially, the idea was to consider a generalization of the notion of inversion, called k-*inversion*, and proving that k-pass sweeping definability is equivalent to asking that every k-inversion generates a periodic output.

We show that we can follow the same approach for two-way transducers. More precisely, we first define a *co-inversion* in a way similar to Definition 14, namely, as a tuple (L_1, C_1, L_2, C_2) consisting of two idempotent loops L_1, L_2, a component C_1 of L_1, and a component C_2 of L_2 such that

- $\mathsf{an}(C_1) \trianglelefteq \mathsf{an}(C_2)$,
- $\mathsf{out}(\mathsf{tr}(C_1)), \mathsf{out}(\mathsf{tr}(C_2)) \neq \varepsilon$, and
- $\mathsf{an}(C_i) = (x_i, y_i)$ for $i = 1, 2$, then $x_1 \leqslant x_2$.

The only difference compared to inversions is the ordering of the positions of the anchors, which is now reversed.

Alternating inversions and co-inversions leads to:

Definition 34. A *k-inversion* is a tuple $\overline{\mathcal{I}} = (\mathcal{I}_0, \ldots, \mathcal{I}_{k-1})$, where $\mathcal{I}_i = (L_i, C_i, L'_i, C'_i)$ is either an inversion or a co-inversion depending on whether i is even or odd, and $\mathsf{an}(C'_i) \trianglelefteq \mathsf{an}(C_{i+1})$ for all $i < k-1$.
A k-inversion $\overline{\mathcal{I}}$ is *safe* if for *some* $0 \leqslant i < k$, the word

$$\mathsf{out}\big(\mathsf{tr}(C_i)\big)\ \mathsf{out}\big(\rho[\mathsf{an}(C_i), \mathsf{an}(C'_i)]\big)\ \mathsf{out}\big(\mathsf{tr}(C'_i)\big)$$

has period $p \leqslant \boldsymbol{B}$ dividing $|\mathsf{out}(\mathsf{tr}(C_i))|$ and $|\mathsf{out}(\mathsf{tr}(C'_i))|$.

Similar to the characterization of k-pass sweeping definability in [4], we show now the following characterization for 2-way transducers, using Theorem 24 as a black-box:

Theorem 35. Let $\mathcal{T}$ be a functional two-way transducer and $k > 0$. The following are equivalent:

1) $\mathcal{T}$ is k-pass sweeping definable.
2) All k-inversions of all successful runs of $\mathcal{T}$ are safe.

The problem of deciding whether the above conditions hold is in 2ExpSpace; more precisely, it can be decided in double exponential space w.r.t. $|\mathcal{T}|$ and in polynomial space w.r.t. k.

Proof sketch. A proof of this result (modulo the necessary changes in complexity due to the new characterization) can be found in [4]. Here we present in an informal way the main steps of the proof.

Proving the implication from 2) to 1) boils down to factorize a successful run ρ of $\mathcal{T}$ into factors $\rho_1, \ldots, \rho_k$ in such a way that, for every odd (resp. even) index i, ρ_i contains only inversions (resp. co-inversions) that are safe, namely, that yield periodic outputs. We use the constructions presented in Section 6 to simulate the output of each factor ρ_i with a one-way transducer, which scans the input either from left to right or from right to left, depending on whether i is odd or even.

The implication from 1) to 2) amounts at showing that every k-inversion is safe under the assumption that $\mathcal{T}$ is k-pass sweeping definable. The proof builds upon the characterization of one-way definability. More precisely, we consider a successful run of $\mathcal{T}$ and the corresponding run of an equivalent k-pass sweeping transducer $\mathcal{T}'$ that produces the same output. We then pump those runs simultaneously on all loops $L_1, \ldots, L_{2k}$ that form the k-inversion. By reasoning as in the proof of Proposition 16, we derive a periodicity property that shows that the k-inversion is safe.

Finally, the 2ExpSpace complexity of the decision problem follows from reducing k-pass sweeping definability to the emptiness of the language $\mathsf{dom}(\mathcal{T}) \cap U^{\complement}$, where U is now the language of words $u \in \mathsf{dom}(\mathcal{T})$ such that all k-inversions of all successful runs on u are safe. As usual the latter problem is solved by constructing an NFA that recognizes $U^{\complement}$ by guessing a successful run ρ of $\mathcal{T}$ and an unsafe k-inversion of ρ. □

A similar problem, called *sweeping definability*, concerns the characterization of those transductions that are definable by sweeping transducers, but this time without enforcing any bound on the number of passes (or reversals). Of course the latter problem is interesting only when the transductions are presented by means of two-way transducers. Below we show that the sweeping definability problem reduces to the k-pass sweeping definability problem, when we set k large enough.

Theorem 36. A functional two-way transducer $\mathcal{T}$ is sweeping definable iff it is k-pass sweeping definable, for $k = 2h_{\mathsf{max}} \cdot (2^{3e_{\mathsf{max}}} + 1)$.

Proof sketch. The right-to-left implication is trivial. For the converse direction we sketch the proof idea. Suppose that $\mathcal{T}$ is not k-pass sweeping definable, for $k = 2h_{\mathsf{max}} \cdot (2^{3e_{\mathsf{max}}} + 1)$. By Theorem 35, there exists a successful run ρ of $\mathcal{T}$ and an unsafe k-inversion $\overline{\mathcal{I}}$ of ρ. One can exploit the fact that k is large enough to find an idempotent loop L and an intercepted factor of it that covers two consecutive (co)inversions of $\overline{\mathcal{I}}$. Then, by pumping the loop L, one can introduce arbitrarily long alternations between inversions and co-inversions, thus showing that there are successful runs with unsafe k'-inversions for all $k' > 0$. By Theorem 35, this proves that $\mathcal{T}$ is not sweeping definable. □

Corollary 37. The problem of deciding sweeping definability of a functional two-way transducer is in 2ExpSpace.

Another consequence is that it is decidable in 2ExpSpace whether a functional two-way transducer is equivalent to some two-way transducer performing a bounded number of reversals in every run. Indeed, in [4] we proved that a functional transducer is k-pass sweeping definable iff it is $(k-1)$-reversal definable.

Other classes of transducers are amenable to characterizations via similar techniques. For example, we may consider an even more restricted variant of transducer, called *rotating transducer*. This is a sweeping transducer that emits output only when moving from left to right. Such a transducer is called *k-pass* if it performs at most k passes from left to right. To characterize those transductions that are definable by k-pass rotating transducers it suffices to modify slightly the definition of k-inversion, by removing co-inversions. Formally, one defines a *rotating k-inversion* as a tuple $\overline{\mathcal{I}} = (\mathcal{I}_0, \ldots, \mathcal{I}_{k-1})$, where each $\mathcal{I}_i = (L_i, C_i, L'_i, C'_i)$ is an inversion and $\mathsf{an}(C'_i) \lhd \mathsf{an}(C_{i+1})$ for all $i < k-1$. The analogous of Theorems 35 and 36 would then carry over.

8. Conclusions

It was shown recently [14] that it is decidable whether a given two-way transducer can be implemented by some one-way transducer, however the complexity of the algorithm is non-elementary.

The main contribution of our paper is a new algorithm that solves the above question with elementary complexity, precisely in 2EXPSPACE. The algorithm is based on a characterization of those transductions, given as two-way transducers, that can be realized by one-way transducers. The flavor of our characterization is different from that of [14]. The approach from [14] is based on a variant of Rabin and Scott's construction [20] of one-way automata, and on local modifications of the two-way run. Our approach relies instead on the global notion of *inversions* and on combinatorial arguments, and is inspired by our previous result for sweeping transducers [3]. The technical challenge in this paper compared to [3] is however significant, and required several involved proof ingredients, ranging from the type of loops we consider, up to the decomposition of the runs.

Our characterization based on inversions yields not only an elementary solution for the problem of one-way definability, but also for definability by sweeping (resp. rotating) transducers, with either known or unknown number of passes. All characterizations above are effective, and can be decided in 2EXPSPACE.

Acknowledgment: The authors would like to thank Ismaël Jecker for fruitful discussions on this topic.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. A general theory of translation. *Math. Syst. Theory*, 3(3):193–221, 1969.

[2] R. Alur and P. Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[3] F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. One-way definability of sweeping transducer. In *FSTTCS*, volume 45 of *LIPIcs*, pages 178–191. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[4] F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. Minimizing resources of sweeping and streaming string transducers. In *ICALP*, volume 55 of *LIPIcs*, pages 114:1–114:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. Full version available at https://hal.archives-ouvertes.fr/hal-01274992.

[5] F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. Untwisting two-way transducers in elementary time, 2017. Full version available at https://arxiv.org/abs/1701.02502.

[6] J. Birget. Two-way automaton computations. *RAIRO - Theoretical Informatics and Applications*, 24(1):47–66, 1990.

[7] O. Carton and L. Dartois. Aperiodic two-way transducers and FO-transductions. In *CSL*, volume 41 of *LIPIcs*, pages 160–174. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[8] C. Choffrut and B. Guillon. An algebraic characterization of unary two-way transducers. In *MFCS*, volume 8634 of *LNCS*, pages 196–207. Springer, 2014.

[9] T. Colcombet. Factorisation forests for infinite words. In *FCT*, volume 4639 of *LNCS*, pages 226–237. Springer, 2007.

[10] L. Daviaud, P. Reynier, and J. Talbot. A generalised twinning property for minimisation of cost register automata. In *LICS*, pages 857–866. ACM, 2016.

[11] S. Eilenberg. *Automata, Langages and Machines*. Academic Press, 1976.

[12] J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, 2(2):216–254, 2001.

[13] E. Filiot, O. Gauwin, and N. Lhote. First-order definability of rational transductions: An algebraic approach. In *LICS*, pages 387–396. ACM, 2016.

[14] E. Filiot, O. Gauwin, P. Reynier, and F. Servais. From two-way to one-way finite state transducers. In *LICS*, pages 468–477. IEEE Computer Society, 2013.

[15] 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-Zentrum für Informatik, 2014.

[16] N. Fine and H. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965.

[17] B. Guillon. Sweeping weakens two-way transducers even with a unary output alphabet. In *NCMA*, volume 318 of *books@ocg.at*, pages 91–108. Österreichische Computer Gesellschaft, 2015.

[18] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[19] J. Kortelainen. On the system of word equations $x_0u_1^ix_1u_2^ix_2\dots u_m^ix_m = y_0v_1^iy_1v_2^iy_2\dots v_m^iy_m$ ($i = 0, 1, 2, \dots$) in a free monoid. *Journal of Automata, Languages and Combinatorics*, 3(1):43–57, 1998.

[20] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.

[21] A. Saarela. Systems of word equations, polynomials and linear algebra: a new approach. *European Journal of Combinatorics*, 47(5):1–14, 2015.

[22] M. Schützenberger. A remark on finite transducers. *Information and Control*, 4(2-3):185–196, 1961.

[23] J. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3(2):198–200, 1959.

[24] I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.