Minimizing resources of sweeping and streaming string transducers*

Félix Baschenis[1], Olivier Gauwin[1], Anca Muscholl[1,2], and Gabriele Puppis[1]

1 **University of Bordeaux, LaBRI, CNRS**
2 **Institute for Advanced Study of the Technical University of Munich**

Abstract

We consider minimization problems for natural parameters of word transducers: the number of passes performed by two-way transducers and the number of registers used by streaming transducers. We show how to compute in ExpSpace the minimum number of passes needed to implement a transduction given as sweeping transducer, and we provide effective constructions of transducers of (worst-case optimal) doubly exponential size. We then consider streaming transducers where concatenations of registers are forbidden in the register updates. Based on a correspondence between the number of passes of sweeping transducers and the number of registers of equivalent concatenation-free streaming transducers, we derive a minimization procedure for the number of registers of concatenation-free streaming transducers.

1998 ACM Subject Classification F.4.3, F.1.1, F.2.0

Keywords and phrases word transducers, streaming, 2-way, sweeping transducers, minimization

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Regular word functions extend the robust family of regular languages, preserving many of its characterizations and algorithmic properties. A word function maps words over a finite input alphabet to words over a finite output alphabet. Regular word functions have been studied in the early seventies, in the form of (deterministic) two-way finite state automata with output [1]. Engelfriet and Hoogeboom [8] later showed that monadic second-order definable graph transductions, restricted to words, are an equivalent model — 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 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. A variant of streaming transducers extended by stacks has been introduced in [3] and shown to capture precisely the monadic-second order definable tree transductions.

Two-way and streaming transducers raise new and challenging questions about storage requirements. The classical storage measure for automata is state complexity. State space minimization of two-way transducers is however poorly understood, even in the simpler setting of automata (cf. related work). But there are more meaningful parameters for

* This work was partially supported by the ExStream project (ANR-13-JS02-0010) and the TUM-IAS, funded by the German Excellence Initiative and the EU 7th Framework Programme (grant 291763).

transducer minimization. One such parameter for streaming transducers is the number of registers, and for two-way transducers it is the number of times the transducer needs to re-process the input word. These parameters measure the required storage capacity in a more realistic way than the number of control states. For example, a two-way transducer that needs to process a very large input with several passes has much larger memory requirements in practice than the memory needed for storing the states. Ideally, the input is processed one-way, hence in one pass only, as in the streaming setting. But not every transduction can be implemented by a one-way, finite state transducer without additional memory.

The register minimization problem has been considered by Alur and Raghothaman in [4], for a special family of deterministic streaming transducers: the output alphabet is unary, and the updates are additions/subtractions of registers by constants. For two-way transducers, Filiot et al. showed how to decide whether a transducer is equivalent to some one-way transducer [10]. The decision procedure of [10] is non-elementary, and we provided in [5] an elementary decision procedure and construction of equivalent one-way transducers in the special case of *sweeping* transducers: head reversals are only allowed at the extremities of the input. Sweeping transducers are strictly less expressive than two-way transducers, as shown e.g. by the transduction mapping inputs of the form $u_1\#u_2\#\cdots\#u_n$, where the words u_i contain no occurrence of $\#$, to $u_n\cdots u_2u_1$.

In this paper we extend our results from [5] by showing how to compute in ExpSpace the *minimal number of passes* needed by a non-deterministic, functional sweeping transducer. It turns out that sweeping transducers have the same expressive power as bounded-reversal two-way transducers, and as *concatenation-free* streaming transducers – transducers where concatenation of registers is not allowed in the updates. Since the transformations between the sweeping and streaming models preserve the relationship between the number of passes and the number of registers, we reduce the *minimization problem for registers* of concatenation-free streaming transducers to the minimization of the number of passes of sweeping transducers, and thus solve the former problem.

Related work. As already mentioned, succinctness questions about two-way automata are still challenging. A longstanding open problem is whether non-deterministic two-way automata are exponentially more succinct than deterministic two-way automata. It is only known that this is the case for deterministic sweeping automata [13].

Regular transductions behave also nicely in terms of expressiveness: first-order definable transductions are known to be equivalent to transductions defined by aperiodic streaming transducers [11] and by aperiodic two-way transducers [6].

Besides [4], the closest work to ours is [7], that shows how to compute the minimal number of registers of deterministic streaming transducers with register updates of the form $x := y \cdot v$, where v is a word and x, y are registers. These transducers are as expressive as one-way transducers. However, the focus of [7] is different from ours, since the outputs can be formed over any infinitary group. Moreover, the works [4, 7] consider deterministic transducers, which require in general more registers than non-deterministic functional ones. The proof techniques are based on variants of a property that has been studied for one-way transducers (the twinning property), and are quite different from ours.

Overview. After introducing two-way and streaming transducers in Section 2, and showing some basic properties, we recall in Section 3 the key characterization of one-way definability from [5]. Section 4 presents the main result on minimization of sweeping transducers. Finally, Section 5 concludes with a logical characterization for sweeping transducers. A longer

version of the paper is available at `https://hal.archives-ouvertes.fr/hal-01274992`.

2 Preliminaries

Here we introduce the transducers we are interested in: two-way and streaming transducers.

Two-way transducers. A *two-way transducer* is a tuple $T = (Q, \Sigma, \Delta, I, E, F)$, where Q is a finite set of states, Σ (resp. Δ) is a finite input (resp. output) alphabet, I (resp. F) is a subset of Q representing the initial (resp. final) states, and $E \subseteq Q \times \Sigma \times \Delta^\star \times Q \times \{\mathsf{left}, \mathsf{right}\}$ is a finite set of transition rules describing, for each state and input symbol, the possible output string, target state, and direction of movement. To enable distinguished transitions at the extremities of the input word, we use two special symbols $\rhd$ and $\lhd$ and assume that the input of a two-way transducer is of the form $u = a_1 \dots a_n$, with $n \geq 2$, $a_1 = \rhd$, $a_n = \lhd$, and $a_i \neq \rhd, \lhd$ for all $i = 2, \dots, n-1$.

Given an input word u, we call *positions* the places between the symbols of u, where the head of a transducer can lie. We can identify the positions of $u = a_1 \dots a_n$ with the numbers $1, \dots, n-1$, where each number x is seen as the position between a_x and a_{x+1}. Since here we deal with *two-way* devices, a position can be visited several times along a run. Formally, we associate the states of the transducer with *locations*, namely, with pairs (x, y), where x is a position and y is a non-negative integer, called *level*. For convenience, we assume that, from a location at even level, the transducer can either move to the next position to the right, without changing the level, or perform a reversal, that is, increment the level by 1 and keep the same position; symmetrically, from a location at odd level, the transducer can either move leftward, without changing the level, or perform a reversal. Locations are ordered according to the following order: $\ell \leq \ell'$ if $\ell = (x, y), \ell' = (x', y')$ and one of the following holds: (1) $y < y'$, or (2) $y = y'$ even and $x \leq x'$, or (3) $y = y'$ odd and $x \geq x'$.

Formally, we define a run on $u = a_1 \dots a_n$ as a sequence of locations, labeled by states and connected by edges, hereafter called *transitions*. The state at a location $\ell = (x, y)$ of a run ρ is denoted $\rho(\ell)$. The transitions must connect pairs of locations $\ell \leq \ell'$ that are either at adjacent positions and on the same level, or at the same position and on adjacent levels. Each transition is labeled with a pair a/v consisting of an input symbol a and a word v produced as output. There are four types of transitions:

$$(x, 2y+1) \xleftarrow{a_{x+1}/v} (x+1, 2y+1) \qquad (x, 2y) \xrightarrow{a_{x+1}/v} (x+1, 2y)$$

$$a_{x+1}/v \circlearrowright \begin{matrix}(x+1, 2y+2)\\(x+1, 2y+1)\end{matrix} \qquad \begin{matrix}(x, 2y+1)\\(x, 2y)\end{matrix} \circlearrowleft a_{x+1}/v$$

The upper left (resp. upper right) transition can occur in a run ρ of T on u provided that $\big(\rho(x+1, 2y+1), a_{x+1}, v, \rho(x, 2y+1), \mathsf{left}\big)$ (resp. $\big(\rho(x, 2y), a_{x+1}, v, \rho(x+1, 2y), \mathsf{right}\big)$) is a valid transition rule of T and a_{x+1} is the $(x+1)$-th symbol of u (assuming that first symbol is $\rhd$). Similarly, the lower left (resp. lower right) transition are called *reversals*, and can occur in a run ρ if $\big(\rho(x+1, 2y+1), a_{x+1}, v, \rho(x+1, 2y+2), \mathsf{right}\big)$ (resp. $\big(\rho(x, 2y), a_{x+1}, v, \rho(x, 2y+1), \mathsf{left}\big)$) is a valid transition rule of T and a_{x+1} is the $(x+1)$-th symbol of u.

We say that a run on $u = a_1 \dots a_n$ is *successful* if it starts with an initial state, either at location $(1, 0)$ or at location $(n-1, 1)$, and ends in a final state, at some location of the form $(1, y_{\mathsf{max}})$ or $(n-1, y_{\mathsf{max}})$. The output produced by a run ρ is the concatenation of the words produced by its transitions, and it is denoted by $\mathsf{out}(\rho)$.

Crossing sequences. An important notion associated with runs of two-way automata is that of crossing sequence. Intuitively, this is a tuple of states that label those locations of a run that visit the same position. Formally, given a successful run ρ of a two-way transducer on input $u = a_1 \dots a_n$, the *crossing sequence of ρ at a position $x \in \{1, \dots, n-1\}$* is the tuple $\rho|x = \big(\rho(x, y_0), \dots, \rho(x, y_h)\big)$, where $y_0 < \dots < y_h$ are all and only the levels of the locations of ρ at position x. The classical transformation of two-way finite state automata into equivalent one-way automata [12] uses crossing sequences.

Properties of two-way transducers. We say that a two-way transducer is

- *sweeping* if every run performs the reversals only at the extremities of the input word, i.e. when reading the symbols $\triangleright$ or $\triangleleft$;
- *L-sweeping* if it is sweeping and all successful runs start at the leftmost location $(1, 0)$;
- *R-sweeping* if it is sweeping and all successful runs start at the rightmost location $(n–1, 1)$;
- *k-pass* if every successful run visits every position of the input at most k times;
- *k-reversal* if every successful run performs at most k reversals;
- *one-way* if it is 1-pass, L-sweeping.

A transducer is *functional* if it produces at most one output on each input. It is called *unambiguous* if it admits at most one successful run on each input. These notions will have the same meaning for streaming transducers, defined later. Clearly, every unambiguous transducer is functional. The converse is not true in general, but we will see later that we can transform the functional transducers considered in this paper so as to enforce unambiguity.

It is easy to see that every unambiguous transducer with n states is $2n$-pass. For functional transducers we can restrict ourselves to considering only *normalized* runs, namely, runs that never visit the same position twice with the same state and the same direction. The reason is that functionality guarantees that every factor of a successful run that starts and ends at the same position and with the same state produces the empty output.

Hereafter, we silently assume that all transducers are *functional* and all successful runs are *normalized*. As a consequence the length of the crossing sequences of the successful runs of a transducer can be bounded by $2n$, where n is the number of states of the transducer.

For streaming transducers, we can observe the following. Every k-pass R-sweeping transducer can be transformed into an equivalent $(k+1)$-pass L-sweeping transducer. It is also easy to disambiguate functional sweeping transducers, that is, transform them into equivalent unambiguous sweeping transducers, without increasing the number of passes. For this it suffices to fix a total order on the successful runs, e.g. the lexicographic order, and restrict to runs that are minimal among those over the same input.

The following proposition shows an interesting correspondence between the number of passes of sweeping transducers and the number of reversals of two-way transducers.

▶ **Proposition 1.** Every k-pass sweeping transducer is also $(k-1)$-reversal. Conversely, every $(k-1)$-reversal two-way transducer can be transformed in 2ExpTime into an equivalent unambiguous k-pass sweeping transducer. The transformation can be performed in ExpTime if the $(k-1)$-reversal transducer is unambiguous.

Streaming transducers. Streaming transducers can implement the same transductions as two-way transducers [2, 8], but they do so using a single left-to-right pass and a fixed set of registers that can store words over the output alphabet.

Formally, a *streaming transducer* is a tuple $T = (Q, \Sigma, \Delta, R, U, I, E, F)$, where Q is a finite set of states, Σ (resp. Δ) is a finite input (resp. output) alphabet, R is a finite set

of registers disjoint from Δ, U is a finite set of *updates* for the registers, namely, functions from R to $(R \uplus \Delta)^\star$, I is a subset of Q representing the initial states, $E \subseteq Q \times \Sigma \times U \times Q$ is a finite set of transition rules, describing, for each state and input symbol, the possible updates and target states, and $F : Q \rightharpoonup (R \uplus \Delta)^\star$ is a partial output function.

A well-behaved class of streaming transducers [2] is obtained by restricting the allowed types of updates and partial output functions to be copyless. A streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$ is *copyless* if (1) for every update $f \in U$, every register $z \in R$ appears at most once in $f(z_1) \cdot \ldots \cdot f(z_k)$, where $R = \{z_1, \ldots, z_k\}$, and (2) for every state $q \in Q$, every register $z \in R$ appears at most once in $F(q)$. Hereafter we assume that all streaming transducers are copyless.

To define the semantics of a streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$, we introduce *valuations* of registers in R. These are functions of the form $g : R \to \Delta^\star$. Valuations can be homomorphically extended to words over $R \cup \Delta$ and to updates, as follows. For every valuation $g : R \to \Delta^\star$ and every word $w \in (R \cup \Delta)^\star$, we let $g(w)$ be the word over Δ obtained from w by replacing every occurrence of a register z with its valuation $g(z)$. Similarly, for every valuation $g : R \to \Delta^\star$ and every update $f : R \to (R \cup \Delta)^\star$, we denote by $g \circ f$ the valuation that maps each register z to the word $g(f(z))$.

A *configuration* of T is a pair state-valuation (q, g). This configuration is said to be initial if $q \in I$ and $g(z) = \varepsilon$ for all registers $z \in R$. When reading a symbol a, the transducer can move from a configuration (q, g) to a configuration (q', g') if there exists a transition rule $(q, a, f, q') \in E$ such that $g' = g \circ f$. We denote this by $(q, g) \xrightarrow[T]{a} (q', g')$.

A *run* of T on $u = a_1 \ldots a_n$ is a sequence of configurations and transitions of the form $\sigma = (q_0, g_0) \xrightarrow[T]{a_1} (q_1, g_1) \xrightarrow[T]{a_2} \ldots \xrightarrow[T]{a_n} (q_n, g_n)$. The run ρ is *successful* if the partial output function F is defined on the last state q_n. In this case, the *output* of T on u is $g_n(F(q_n))$.

Properties and relationships with sweeping transducers. Functional and unambiguous streaming transducers are defined as in the two-way case. A streaming transducer is *k-register* if it uses at most k registers. As we did for two-way transducers, we assume that all streaming transducers are functional.

It is known that (functional) streaming transducers capture precisely the transductions definable by deterministic two-way transducers or, equally, by monadic second-order logic (so-called MSO transductions) [2, 8]. Moreover, differently from two-way transducers, nondeterministic streaming transducers can be determinized. This happens at the cost of increasing the number of registers.

▶ **Definition 2.** A streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$ is *concatenation-free* if $f(z) \in \Delta^\star \cdot (R \cup \{\varepsilon\}) \cdot \Delta^\star$, for all registers $z \in R$ and all updates $f \in U$.

Intuitively, a concatenation-free streaming transducer forbids register updates with two or more registers inside a right-hand side. We note that concatenation-free streaming transducers can also be determinized effectively. Moreover, it is easy to see that allowing boundedly many updates with concatenations does not change the expressiveness of the model, as one can remove any occurrence of an update with concatenations by introducing new registers.

The following proposition shows a tight correspondence between the number of registers of the concatenation-free streaming transducers and the number of passes of the sweeping transducers. Note that the proposition considers sweeping transducers that start from the rightmost position. A slightly weaker correspondence holds for L-sweeping transducers, since any sweeping transducer can be made L-sweeping (resp. R-sweeping) by increasing the number of passes by 1.

▶ **Proposition 3.** Every concatenation-free streaming transducer with k registers can be transformed in ExpTime into an equivalent unambiguous $2k$-pass R-sweeping transducer. The transformation is in Ptime if the streaming transducer is unambiguous.

Conversely, every k-pass R-sweeping transducer can be transformed in 2ExpTime into an equivalent unambiguous concatenation-free streaming transducer with $\lceil \frac{k}{2} \rceil$ registers. The transformation is in ExpTime if the sweeping transducer is unambiguous.

Based on the above proposition, the problem of minimizing the number of registers in a concatenation-free streaming transducer reduces to the problem of minimizing the number of passes performed by a sweeping transducer. We will thus focus on the latter problem: in Section 4, we consider the decidability and complexity of the following problem, called *k-pass sweeping definability problem*: given a functional sweeping transducer S and a number $k \in \mathbb{N}$, decide whether S has an equivalent k-pass sweeping transducer.

3 One-way definability

In [5] we gave an effective characterization of sweeping transducers that are *one-way definable*, i.e., equivalent to some one-way transducer. This can be seen as a special case of the problem that we are considering here, and some of the technical tools developed in [5] will be used later. We briefly recall some definitions and results related to this characterization. Hereafter we assume that S is an L-sweeping transducer and ρ a successful run of S.

Intercepted factors. An *interval* of positions of the run ρ has the form $I = [x_1, x_2]$, with $x_1 < x_2$. We say that an interval $I = [x_1, x_2]$ *contains* (resp., *strongly contains*) another interval $I' = [x'_1, x'_2]$ if $x_1 \leq x'_1 \leq x'_2 \leq x_2$ (resp., $x_1 < x'_1 \leq x'_2 < x_2$). We say that a factor of ρ is *intercepted* by an interval $I = [x_1, x_2]$ if it is maximal among the factors of ρ that visit only positions in I and that never make a reversal (recall that reversals in sweeping transducers can only occur at the extremities of the input word).

Pumping loops. A *loop* of a run ρ is an interval $L = [x_1, x_2]$ of positions such that the crossing sequences $\rho|x_1$ and $\rho|x_2$ are equal. If L is a loop of ρ, we can obtain new runs by replicating any number of times the factors of ρ intercepted by L and, simultaneously, the factor of the input word u between positions x_1 and x_2. This operation is called *pumping* and is formally defined as follows. Let $L = [x_1, x_2]$ be a loop of a run ρ on u. The run obtained by pumping n times the loop L is the sequence $\mathsf{pump}^n_L(\rho) = \underbrace{\alpha_1\, \boldsymbol{\beta_1^n}\, \gamma_1}_{\text{1st pass}}\ \underbrace{\alpha_2\, \boldsymbol{\beta_2^n}\, \gamma_2}_{\text{2nd pass}}\ \cdots\ \underbrace{\alpha_k\, \boldsymbol{\beta_k^n}\, \gamma_h}_{k\text{-th pass}}$ where k is the number of passes performed by ρ, β_i is the factor intercepted by L at the i-th level, α_i is the factor intercepted either by $[1, x_1]$ or by $[x_2+1, |u|]$ at the i-th level, depending on whether this level is even or odd, and, symmetrically, γ_i is the factor intercepted either by $[x_2 + 1, |u|]$ or by $[1, x_1]$ at the i-th level, depending on whether this level is even or odd. We also define $\mathsf{pump}^n_L(u) = u[1, x_1] \cdot \boldsymbol{\big(u[x_1 + 1, x_2]\big)^n} \cdot u[x_2 + 1, |u|]$ and we observe that $\mathsf{pump}^n_L(\rho)$ is a valid run on $\mathsf{pump}^n_L(u)$.

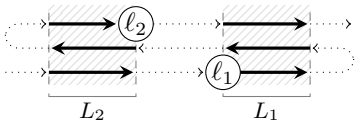
It is convenient to introduce some notation for pumping runs on multiple loops. If the loops are pairwise non-overlapping this can be done by simply pumping each loop separately, since the order in which we pump the loops does not really matter. The situation is a bit more complicated when some loops overlap. In particular, when pumping a loop L of ρ, several copies of the original locations of ρ are introduced, and with this several copies of other loops may appear (think, for example, of a loop L' that is contained in L). We say that a location $\tilde{\ell}$ in $\mathsf{pump}^n_L(\rho)$ *corresponds* to ℓ in ρ if $\tilde{\ell}$ is one of the copies of ℓ that is introduced

when pumping ρ on L. We extend this correspondence to sets of locations and loops. With a slight abuse of notation, we denote by $\mathsf{pump}^{n_2}_{L_2}(\mathsf{pump}^{n_1}_{L_1}(\rho))$ the run obtained by first pumping n_1 times the loop L_1 in ρ, and then pumping n_2 times *every loop* that corresponds to L_2 in $\mathsf{pump}^{n_1}_{L_1}(\rho)$ (note that the copies of L_2 in $\mathsf{pump}^{n_1}_{L_1}(\rho)$ are pairwise non-overlapping). It is routine to check that the two runs $\mathsf{pump}^{n_2}_{L_2}(\mathsf{pump}^{n_1}_{L_1}(\rho))$ and $\mathsf{pump}^{n_1}_{L_1}(\mathsf{pump}^{n_2}_{L_2}(\rho))$ are isomorphic. This allows us to use the shorthand $\mathsf{pump}^{\overline{n}}_{\overline{L}}(\rho)$ to denote runs obtained from ρ by pumping the loops $\overline{L} = L_1, \ldots, L_m$ with the numbers $\overline{n} = n_1, \ldots, n_m$, respectively.

Inversions. The notion of inversion is crucial for characterizing one-way definability [5]. Let L be a loop of ρ. A location ℓ_1 is called an *entry point of* L if it is the first location of a factor intercepted by L. Similarly, a location ℓ_2 is called an *exit point of* L if it is the last location of a factor intercepted by L. Note that every entry/exit point of $L = [x_1, x_2]$ occurs either at position x_1 or at position x_2.

▶ **Definition 4.** An *inversion* of a run ρ is a pair of locations ℓ_1 and ℓ_2 for which there exist two loops $L_1 = [x_1, x_1']$ and $L_2 = [x_2, x_2']$ such that (also refer to the figure on the right):

- ℓ_1 is an entry point of L_1 and ℓ_2 is an exit point of L_2,
- $\ell_1 < \ell_2$ and $x_2 \leq x_1'$,
- for both $i = 1$ and $i = 2$, the factor intercepted by L_i and visiting ℓ_i has non-empty output, and no other loop strongly contained in L_i has the same property as L_i w.r.t. this factor.



We say that the loops L_1 and L_2 are the *witnessing loops* of the inversion (ℓ_1, ℓ_2).

Periodic words. A word w is said to have *period* p if $w \in u^* v$ for some word u of length p and some prefix v of u. For example, $w = abcabcab$ has period $p = 3$.

We are interested into factors of the outputs of S that are periodic, with uniformly bounded periods. To do this, we fix the constant $e_S = c_S \cdot |Q|^{2|Q|}$, where c_S is the maximum number of symbols output by a single transition of S and Q is the state space of S. The crux in [5] is the following property:

▶ **Proposition 5** (Prop. 7 in [5])**.** If S is a one-way definable L-sweeping transducer and (ℓ_1, ℓ_2) is an inversion of a successful run ρ of S, then $\mathsf{out}(\rho[\ell_1, \ell_2])$ has period at most e_S.

The above result justifies the following definition: let $L_S \subseteq \mathrm{dom}(S)$ be the language of those words u that induce a successful run ρ of S such that, for all inversions (ℓ_1, ℓ_2) of ρ, $\mathsf{out}(\rho[\ell_1, \ell_2])$ is periodic with period at most e_S. We denote by $S|_{L_S}$ the transducer S restricted to inputs from L_S. One-way definability is characterized as follows:

▶ **Theorem 6** (Th. 1 in [5])**.** *An L-sweeping transducer S is one-way definable if and only if $L_S = dom(S)$. Moreover, given an L-sweeping transducer S, one can construct in doubly exponential time a one-way transducer T that is equivalent to $S|_{L_S}$.*

4 k-pass sweeping definability

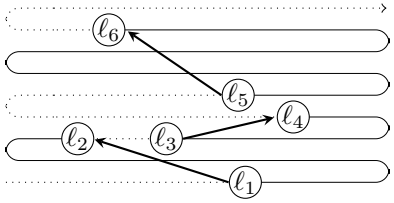
We begin by defining the objects that need to be considered for characterizing *k-pass definability*, i.e., whether a sweeping transducer is equivalent to some k-pass sweeping transducer. Let S be an L-sweeping transducer. The idea is to consider factors of runs of S that can be simulated alternatively from left to right and from right to left.We begin by introducing a notion of inversion that looks symmetric to Definition 4: a *co-inversion* is defined as above, with $x_1 \leq x_2'$ replacing $x_2 \leq x_1'$. In other words, for an inversion we exclude the case where

L_2 is *after* L_1, whereas for a co-inversion we exclude that L_2 is *before* L_1. We then combine inversions and co-inversions, as follows:

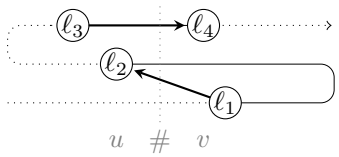
▶ **Definition 7.** A *k-inversion* of ρ is a sequence $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ such that:

- $\ell_1 < \ell_2 < \ldots < \ell_{2k-1} < \ell_{2k}$ are distinct locations in ρ,
- for all even $i \in \{0, \ldots, k-1\}$, $(\ell_{2i+1}, \ell_{2i+2})$ is an inversion of ρ,
- for all odd $i \in \{0, \ldots, k-1\}$, $(\ell_{2i+1}, \ell_{2i+2})$ is a co-inversion of ρ.

An example of a 3-inversion is depicted to the right. We say that $\overline{\ell}$ is *safe* if $\mathsf{out}(\rho[\ell_{2i+1}, \rho_{2i+2}])$ has period at most e_S, for *some* $i \in \{0, \ldots, k-1\}$. We denote by $L_S^{(k)}$ the language of words $u \in \mathrm{dom}(S)$ such that all k-inversions of all successful runs of S on u are safe.



▶ **Example 8.** Consider the 3-pass transducer that on input $u\#v$, with $u, v \in \{a, b\}^*$, outputs $(ab)^{|uvv|}(ba)^{|uuv|}$. This transduction can also be realized in 2 passes. This means that every 2-inversion is safe. For example, the 2-inversion depicted to the right is safe, as the output $\rho[\ell_3, \ell_4]$ is periodic.



Note that the definition of 1-inversion is the same as Definition 4, and hence $L_S^{(1)} = L_S$. In particular, by Theorem 6, we know that S is one-way definable iff $L_S^{(1)} = \mathrm{dom}(S)$. The generalization of this result is provided in Theorem 9 below: k-pass definability is equivalent to k-inversions being all safe, in the same way as one-way definability is equivalent to all inversions having periodic output.

▶ **Theorem 9.** *A sweeping transducer S is k-pass L-sweeping definable iff $L_S^{(k)} = dom(S)$, and this can be decided in* ExpSpace*. Moreover, given a sweeping transducer S, one can construct in* 2ExpTime *an unambiguous k-pass L-sweeping transducer T equivalent to $S|_{L_S^{(k)}}$.*

An analogous result for deciding k-pass *R-sweeping* definability can be derived by symmetry, by mirroring the input and reversing the computation. We also observe that, for $k = 1$, the above theorem improves the previous 2ExpSpace upper bound from [5] for deciding one-way definability of a sweeping transducer S. Concerning the doubly exponential size of an equivalent k-pass L-sweeping transducer, we observe that this is optimal, as in [5] we have shown that there are sweeping transducers S such that any equivalent one-way transducer has size at least doubly exponential in S.

Before turning to the proof of Theorem 9, we list some simple consequences of this theorem and of Propositions 1 and 3.

▶ **Corollary 10.**

- *One can compute in* ExpSpace *the minimum number of passes needed to implement a transduction given as a sweeping transducer.*
- *One can compute in* 3ExpSpace *the minimum number of reversals needed to implement a transduction given as a bounded-reversal two-way transducer. The complexity is* 2ExpSpace *if the given two-way transducer is unambiguous.*
- *One can compute in* 2ExpSpace *the minimum number of registers needed to implement a transduction given as a concatenation-free streaming transducer. The complexity is* ExpSpace *if the given streaming transducer is unambiguous.*

The proof of Theorem 9 is split into two parts. The first part, called "soundness", deals with the construction of the k-pass L-sweeping transducer T of the second claim.

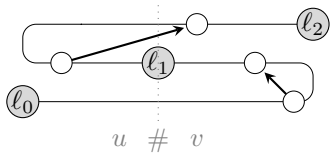
Since $L_S^{(k)} = \mathrm{dom}(S)$ implies that T is equivalent to S, this construction also proves the right-to-left direction of the first claim. Moreover, as a side result, we prove that whether $L_S^{(k)} = \mathrm{dom}(S)$ holds is decidable in ExpSpace. The second part, called "completeness", deals with the left-to-right direction of the first claim.

Soundness. We show how to construct from S a k-pass L-sweeping transducer T equivalent to $S|_{L_S^{(k)}}$. The idea is to consider a successful run ρ of S on a word $u \in L_S^{(k)}$, and divide it into k factors. We then simulate each factor of the run in a single pass, alternatively from left to right and from right to left, using [5]. First we need the notion of *k-factorizations*:

▶ **Definition 11.** A *k-factorization* of a successful run ρ of S is any sequence of locations $\overline{\ell} = \ell_0, \ell_1, \ldots, \ell_k$ of ρ such that:

- $\ell_0 \leq \ell_1 \leq \ldots \leq \ell_k$, ℓ_0 is the first location of ρ, and ℓ_k is the last location of ρ,
- for all even indexes i, with $0 \leq i < k$, and all inversions (ℓ, ℓ') of ρ, with $\ell_i \leq \ell \leq \ell' \leq \ell_{i+1}$, the word $\mathsf{out}(\rho[\ell, \ell'])$ has period at most e_S,
- for all odd indexes i, with $1 \leq i < k$, and all co-inversions (ℓ, ℓ') of ρ, with $\ell_i \leq \ell \leq \ell' \leq \ell_{i+1}$, the word $\mathsf{out}(\rho[\ell, \ell'])$ has period at most e_S.

▶ **Example 12.** We consider the transducer of Example 8 and we depict a 2-factorization of a run of it (gray nodes). All the inversions between ℓ_0 and ℓ_1, and all the co-inversions between ℓ_1 and ℓ_2, must be periodic. Note that the run does contain non-periodic inversions (e.g. those crossing ℓ_1), and hence it does not admit a 1-factorization.



The following lemma shows that we can reason equally in terms of safe k-inversions (Definition 7) and in terms of k-factorizations.

▶ **Lemma 13.** *For every word $u \in dom(S)$, we have that $u \in L_S^{(k)}$ if and only if all successful runs of S on u admit k-factorizations.*

Next we show that being a k-factorization is a *regular* property. To formalize this, we need to explain how to encode runs and sequences of locations as annotations of the underlying input. Formally, given a word $u \in \mathrm{dom}(S)$, a successful run ρ of S on u, and a tuple of locations $\overline{\ell} = \ell_1, \ldots, \ell_m$ in ρ, we denote by $\langle u, \rho, \overline{\ell} \rangle$ the word obtained by annotating each position $1 \leq x < |u|$ of u with the crossing sequence $\rho|x$ and with the m-tuple $\overline{y} = (y_1(x), \ldots, y_m(x))$, where each $y_i(x)$ is either the level of ℓ_i or $\perp$, depending on whether ℓ_i is at position x or not. Based on this encoding, we can define the language $F_S^{(k)}$ of all words of the form $\langle u, \rho, \overline{\ell} \rangle$, where ρ is a successful run of S on u and $\overline{\ell} = \ell_0, \ldots, \ell_k$ is a k-factorization of ρ. Lemma 14 below proves that this language is regular. In fact, in order to better handle the complexity of our characterization, the lemma shows that both $F_S^{(k)}$ and its complement $\overline{F}_S^{(k)}$ are recognized by automata of doubly exponential size.

▶ **Lemma 14.** *The language $F_S^{(k)}$ and its complement $\overline{F}_S^{(k)}$ are recognized by non-deterministic finite state automata of size double exponential w.r.t. S.*

Using the above encodings, we can also relativize the outputs produced by the transducer S to factors of successful runs. More precisely, we denote by S_{factors} the transducer that reads words of the form $\langle u, \rho, \ell_1, \ell_2 \rangle$ and outputs words of the form $\mathsf{out}(\rho[\ell_1, \ell_2])$, provided that ρ is a successful run of S on u and ℓ_1, ℓ_2 are two locations in it. Note that S_{factors} does

not check that the input is well-formed, in particular, that ρ is a successful run of S on u. Because of this, the number of states of S_{factors} is polynomial in the number of states of S, and a succinct representation of S_{factors} can be produced in polynomial time.

Now, it is easy to construct a k-pass L-sweeping transducer T equivalent to $S|_{L_S^{(k)}}$, as claimed in Theorem 9. The idea is that, on reading the input u, the transducer T guesses a successful run ρ on u and a k-factorization $\overline{\ell} = \ell_0, \ldots, \ell_k$ of ρ — this can be done using the encoding $\langle u, \rho, \overline{\ell}\rangle$ and Lemma 14. While guessing these objects, T performs k passes and outputs $T_0(\langle u, \rho, \ell_0, \ell_1\rangle) \cdot T_1(\langle u, \rho, \ell_1, \ell_2\rangle) \cdot \ldots \cdot T_{k-1}(\langle u, \rho, \ell_{k-1}, \ell_k\rangle)$, where each T_i is the 1-pass sweeping transducer obtained by applying Theorem 6 to S_{factor} (as usual, some mirroring is required for dealing with the odd indexes i). The only technical detail, here, is that different objects $\rho, \overline{\ell}$ may be guessed along the different passes of T. If this happens, the output produced by T might not be equal to that of S. We can overcome this problem by exploiting disambiguation, namely, by guessing canonical encodings $\langle u, \rho, \overline{\ell}\rangle$ in the language $F_S^{(k)}$. For example, we can fix a lexicographic ordering on these encodings and commit to always guessing the least encoding among those that agree on the input word u. This requires reasoning with both the language $F_S^{(k)}$ and its complement $\overline{F}_S^{(k)}$. By Lemma 14, the two languages are recognized by automata of doubly exponential size in S, and hence T can be constructed in doubly exponential time from S. As a matter of fact, the transducer T that we just constructed is also unambiguous.

We conclude this part by showing how to decide in exponential space if $L_S^{(k)} = \text{dom}(S)$. In fact, as we already know that $L_S^{(k)} \subseteq \text{dom}(S)$, it suffices to decide only the containment $L_S^{(k)} \supseteq \text{dom}(S)$. We know from Lemma 13 that the language $L_S^{(k)}$ coincides with the projection of $F_S^{(k)}$ on the underlying words u. Thus, we have $L_S^{(k)} \supseteq \text{dom}(S)$ if and only if $\overline{F}_S^{(k)} \cap D = \emptyset$, where $D = \{\langle u, \rho, \overline{\ell}\rangle \ : \ u \in \text{dom}(S)\}$. A close inspection of the construction of the automaton for $\overline{F}_S^{(k)}$ shows that the emptiness of $\overline{F}_S^{(k)} \cap D$ can be decided in ExpSpace.

Completeness. Here we prove the left-to-right direction of the first claim of Theorem 9. We suppose that S is an L-sweeping transducer and T is an equivalent k-pass L-sweeping transducer. We fix, once and for all, a successful run ρ of S on u and a k-inversion $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ of ρ.

The goal is to prove that $\overline{\ell}$ is safe, namely, that the factor of the output produced between the locations of some (co-)inversion $(\ell_{2i+1}, \ell_{2i+2})$ of $\overline{\ell}$ is periodic, with uniformly bounded period. The main idea is to try to find a factor $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ that is entirely covered by the output produced along a single pass of the equivalent transducer T, and apply a suitable generalization of Proposition 5. Informally, this works by pumping the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ through repeating the witnessing loops of $(\ell_{2i+1}, \ell_{2i+2})$. In a similar way, we pump the output produced along a single pass of T. Then, by analyzing how the former outputs are covered by the latter outputs, we deduce the periodicity of $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$.

The main difficulty in formalizing the above idea lies in the fact that the k passes of the supposed transducer T cannot be identified directly on the run ρ of S. Therefore we need to reason in a proper way about *families* of factors associated with (co-)inversions inside pumped runs. Below, we introduce some terminology and notation to ease this task.

Recall that $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ is a k-inversion of the run ρ. For all $i = 0, \ldots, k-1$, let L_{2i+1} and L_{2i+2} be the witnessing loops of $(\ell_{2i+1}, \ell_{2i+2})$. For a given tuple of numbers $\overline{n} = (n_1, \ldots, n_{2k}) \in \mathbb{N}^{2k}$, we define $\rho^{\overline{n}} = \mathsf{pump}_{\overline{L}}^{\overline{n}}(\rho)$, where $\overline{L} = L_1, \ldots, L_{2k}$ and $\overline{n} = n_1, \ldots, n_{2k}$ (recall that this is the run obtained by pumping the loops $L_1, \ldots, L_{2k}$ respectively $n_1, \ldots, n_{2k}$ times, as described in Section 3). Similarly, we denote by $u^{\overline{n}}$ the word parsed

by the pumped run $\rho^{\overline{n}}$.

We would like to map the inversions and co-inversions of $\overline{\ell}$ on the pumped runs $\rho^{\overline{n}}$. Consider an inversion $(\ell_{2i+1}, \ell_{2i+2})$, for some $i \in \{1, \dots, 2k\}$ (the case of a co-inversion is similar). Recall that when pumping loops in ρ, several copies of the original locations may be introduced. In particular, among the copies of the inversion $(\ell_{2i+1}, \ell_{2i+2})$ that appear in the pumped run $\rho^{\overline{n}}$, we will consider the maximal one, which is identified by taking the *first* copy $\tilde{\ell}_{2i+1}$ of ℓ_{2i+1} and the *last* copy $\tilde{\ell}_{2i+2}$ of ℓ_{2i+2}. For the sake of brevity, we say that $(\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2})$ is the inversion of $\rho^{\overline{n}}$ that *corresponds* to $(\ell_{2i+1}, \ell_{2i+2})$.

We can now define the key objects for our reasoning, that is, the factors of the output of a pumped run $\rho^{\overline{n}}$ that correspond in the original run ρ to the factors produced between the locations of the (co-)inversions of $\overline{\ell}$. Formally, for every $2k$-tuple $\overline{n}$ of natural numbers and every index $i = 0, \dots, k-1$, we define

$$v^{\overline{n}}(i) \;=\; \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2}]\big)$$

where $(\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2})$ is the (co-)inversion of $\rho^{\overline{n}}$ that corresponds to $(\ell_{2i+1}, \ell_{2i+2})$. Below we highlight the relevant factors inside the output produced by S on $u^{\overline{n}}$:

$$S(u^{\overline{n}}) \;=\; \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_0, \tilde{\ell}_1]\big) \cdot \boldsymbol{v^{\overline{n}}(0)} \cdot \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_2, \tilde{\ell}_3]\big) \cdot \boldsymbol{v^{\overline{n}}(1)} \cdot \ldots \cdot \boldsymbol{v^{\overline{n}}(k-1)} \cdot \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_{2k}, \tilde{\ell}_{2k+1}]\big) \quad (1)$$

where $\tilde{\ell}_0$ is the first location of $\rho^{\overline{n}}$, $\tilde{\ell}_{2k+1}$ is the last location of $\rho^{\overline{n}}$.

In a similar way, we can factorize the output produced by the k-pass L-sweeping transducer T when reading the input $u^{\overline{n}}$. However, the focus here is on the factors of the output produced along each pass. Formally, given $\overline{n} \in \mathbb{N}^{2k}$, we let $\sigma^{\overline{n}}$ be some successful run of T on $u^{\overline{n}}$. For every $j = 0, \dots, k-1$, we let ℓ'_j be the first location of $\sigma^{\overline{n}}$ at level j. We further let ℓ'_k be the last location of $\sigma^{\overline{n}}$, which is at level $k-1$. We then define

$$w^{\overline{n}}(j) \;=\; \mathsf{out}\big(\sigma^{\overline{n}}[\ell'_j, \ell'_{j+1}]\big)$$

and factorize the output of T on $u^{\overline{n}}$ as follows:

$$T(u^{\overline{n}}) \;=\; \boldsymbol{w^{\overline{n}}(0)} \cdot \boldsymbol{w^{\overline{n}}(1)} \cdot \ldots \cdot \boldsymbol{w^{\overline{n}}(k-1)}. \quad (2)$$

The next step is to exploit the hypothesis that S and T are equivalent. This means that Equations (1) and (2) represent the same word. From this we derive that, for any given $\overline{n} \in \mathbb{N}^{2k}$, at least one of the words $v^{\overline{n}}(i)$ highlighted in Equation (1) is a factor of the word $w^{\overline{n}}(i)$ highlighted in Equation (2). However, what is the index i for which this coverability relation holds depends on the parameter $\overline{n}$. In order to enable a reasoning similar to that of Proposition 5, we need to find a single index i such that, for "sufficiently many" parameters $\overline{n}$, $v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$. The definition below, formalizes what we mean precisely by "sufficiently many" $\overline{n}$ — intuitively, we require that specific coordinates of $\overline{n}$ are unbounded, as well the differences between these coordinates.

▶ **Definition 15.** Let $\mathcal{P}(\overline{n})$ denote an arbitrary property of tuples $\overline{n} \in \mathbb{N}^{2k}$. Further let h, h' be two distinct coordinates in $\{1, \dots, 2k\}$. We say that $\mathcal{P}(\overline{n})$ holds *unboundedly on the coordinates h, h' of $\overline{n}$* if, for all numbers $n_0 \in \mathbb{N}$, there exist $\overline{n}_1, \overline{n}_2 \in \mathbb{N}^{2k}$ such that:

- $\mathcal{P}(\overline{n}_1)$ and $\mathcal{P}(\overline{n}_2)$ hold,
- $\overline{n}_1[h] \geq n_0$ and $\overline{n}_1[h'] - \overline{n}_1[h] \geq n_0$,
- $\overline{n}_2[h'] \geq n_0$ and $\overline{n}_2[h] - \overline{n}_2[h'] \geq n_0$.

We recall that each factor $v^{\overline{n}}(i)$ is associated with the (co-)inversion $(\ell_{2i+1}, \ell_{2i+2})$, and that the corresponding components $\overline{n}[2i+1]$ and $\overline{n}[2i+2]$ of the parameter $\overline{n}$ denote the number of times the witnessing loops L_{2i+1} and L_{2i+2} are pumped in $\rho^{\overline{n}}$. The specific properties we are interested in are the following ones, for $i = 0, \ldots, k-1$:

$\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ *is a factor of* $w^{\overline{n}}(i)$".

It is not difficult to see that for every tuple $\overline{n} \in \mathbb{N}^{2k}$, $\mathcal{P}_i(\overline{n})$ holds for some $i \in \{0, \ldots, k-1\}$. From this, using a suitable counting argument, we can prove the crucial lemma below.

▶ **Lemma 16.** *There exists an index* $i \in \{0, \ldots, k-1\}$ *such that the property* $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$" *holds unboundedly on the coordinates* $2i+1$ *and* $2i+2$ *of* $\overline{n}$.

The last piece of the puzzle consists of generalizing the statement of Proposition 5. The idea is that we can replace the hypothesis that S is one-way definable by the weaker assumption of Lemma 16. That is, if $\mathcal{P}_i(\overline{n})$ holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$, we can still use the same arguments based on pumping and Fine-Wilf's Theorem as in Proposition 5, in order to deduce that the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ between the locations of the (co-)inversion is periodic:

▶ **Proposition 17.** If the property $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ *is a factor of* $w^{\overline{n}}(i)$ " holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$, then the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ produced between the locations of the (co-)inversion $(\ell_{2i+1}, \ell_{2i+2})$ is periodic, with period e_S. In particular, the k-inversion $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ is safe.

To conclude, we assumed that the L-sweeping transducer S is equivalent to a k-pass L-sweeping transducer T. We considered a successful run ρ of S and an arbitrary k-inversion $\overline{\ell}$ of it. By Lemma 16, we know that there is an index $i \in \{0, \ldots, k-1\}$ for which the property $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ *is a factor of* $w^{\overline{n}}(i)$ " holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$. From this, by applying Proposition 17, we derive that the k-inversion $\overline{\ell}$ is safe. This proves the left-to-right direction of the first claim of Theorem 9. ◀

5 Sweeping transducers and MSO

We provide a logical characterization of sweeping transducers. For this we will consider restricted forms of transductions definable in monadic-second order logic (MSO) [8].

MSO transductions are described by specifying the output (seen as a relational structure) from a fixed number of copies of the input. Formally, an *MSO transduction with* m *copies* consists of an MSO sentence Φ_{dom}, some unary MSO formulas $\Phi^i_a(x)$, one for each $i \in \{1, \ldots, m\}$ and $a \in \Delta$, and some binary MSO formulas $\Phi^{i,j}_{\leq}(x, y)$, one for each $i, j \in \{1, \ldots, m\}$. Intuitively, the sentence Φ_{dom} tells whether the transduction is defined on some input u. The unary formula $\Phi^i_a(x)$ tells whether the element x of the i-th copy of the input belongs to the output and is labeled with the letter a. The formula $\Phi^{i,j}_{\leq}(x, y)$ tells whether, in the produced output, the element x of the i-th copy of the input precedes the element y of the j-th copy of the input. Note that the sentence Φ_{dom} can easily guarantee that, whenever the output is defined, it is well-formed, namely, it is a word. For the sake of simplicity, we assume that $\Phi^i_a(x)$ *entails* Φ_{dom}, namely, for all words u and all positions x, $u \models \Phi^i_a(x)$ implies $u \models \Phi_{\mathrm{dom}}$. Similarly, we assume that $\Phi^{i,j}_{\leq}(x, y)$ entails $\Phi^i(x)$ and $\Phi^j(y)$.

▶ **Definition 18.** Let T be an MSO transduction with m copies. We say that T is

- *order-preserving* if each formula $\Phi^{i,j}_{\leq}(x, y)$ entails $x \leq y$;
- *order-inversing* if each formula $\Phi^{i,j}_{\leq}(x, y)$ entails $x \geq y$;

- *k-phase* if there is a partition $I_0, I_1, \ldots, I_{k-1}$ of the set of indexes $\{1, \ldots, m\}$ such that $I_0 < I_1 < \ldots < I_{k-1}$, namely, $i < j$ for all $0 \leq h < h' < k$, $i \in I_h$, and $j \in I_{h'}$, and each formula $\Phi^{i,j}_{\leq}(x, y)$ entails $x \leq y$ if h is even, or $x \geq y$ otherwise.

We know from [9] that order-preserving MSO transductions capture precisely the one-way definable transductions. We obtain:

▶ **Theorem 19.** *k-phase MSO transductions have the same expressive power as functional, k-pass L-sweeping transducers.*

6 Conclusions

We showed that sweeping transducers, bounded-reversal transducers, and concatenation-free streaming transducers define the same subclass of regular word transductions. Our main result is an effective characterization of transductions definable by sweeping transducers with a fixed number of passes. As a consequence we obtained a procedure that minimizes the number of registers in a concatenation-free sweeping transducer.

We believe that similar results can be proven for two-way (non-sweeping) transducers, using a refined version of the constructions presented here. In this respect, an interesting open problem is to characterize the two-way transducers that are equivalent to sweeping transducers, but with an arbitrary (unspecified) number of passes.

References

1. A. V. Aho and J. D. Ullman. A characterization of two-way deterministic classes of languages. *J. Comput. Syst. Sci.*, 4(6):523–538, 1970.
2. R. Alur and P. Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
3. R. Alur and L. D'Antoni. Streaming tree transducers. In *ICALP*, volume 7392 of *LNCS*, pages 42–53. Springer, 2012.
4. R. Alur and M. Raghothaman. Decision problems for additive regular functions. In *ICALP*, volume 7966 of *LNCS*, pages 37–48. Springer, 2013.
5. F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. One-way definability of sweeping transducers. In *FSTTCS*, volume 45 of *LIPIcs*, pages 178–191. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
6. O. Carton and L. Dartois. Aperiodic two-way transducers and FO-transductions. In *CSL*, LIPIcs, pages 160–174. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.
7. L. Daviaud, P.-A. Reynier, and J.-M. Talbot. A generalised twinning property for minimisation of cost register automata. In *LICS*. IEEE Computer Society, 2016.
8. J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, 2:216–254, 2001.
9. E. Filiot. Logic-automata connections for transformations. In *ICLA*, pages 30–57, 2015.
10. E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. From two-way to one-way finite state transducers. In *LICS*, pages 468–477. IEEE Computer Society, 2013.
11. 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 fuer Informatik, 2014.
12. J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.
13. M. Sipser. Lower bounds on the size of sweeping automata. In *STOC*, pages 360–364. ACM, 1979.