Minimizing resources of sweeping and streaming string transducers *

Félix Baschenis, Olivier Gauwin, Anca Muscholl, Gabriele Puppis

Minimizing resources of sweeping and streaming string transducers*

Félix Baschenis, Olivier Gauwin, Anca Muscholl†, Gabriele Puppis

Université de Bordeaux, LaBRI, CNRS

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.

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 one-way transducer model called streaming transducer, and showed that it is equivalent to the two previous ones. 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 with 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 notion for automata is state complexity. But state space minimization of two-way transducers is still poorly understood, even in the simpler setting of two-way finite automata (cf. related work).

There are other meaningful parameters that can be used in the minimization of streaming transducers and two-way transducers, respectively. For streaming transducers it 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 an often more realistic way than the number of control states. For example, a two-way transducer that needs to process a potentially 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.

* This work was partially supported by the ExStream project (ANR-13-JS02-0010).
† On leave at the Institute for Advanced Studies of the Technical University of Munich, the support of which is kindly acknowledged.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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/substractions of registers by constants. This model is known as cost register automata. 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 obtained recently [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.

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 the class of sweeping transducers that we consider here has the same expressive power as the class of bounded-reversal two-way transducers. In addition, we show a tight connection between sweeping transducers and streaming transducers: the former are equivalent to concatenation-free streaming transducers, i.e., transducers where concatenation of registers is not allowed in the updates (but may be used in the output). Our transformations preserve the relationship between the number of passes and the number of registers. This allows us to reduce the *minimization problem for registers* of concatenation-free streaming transducers to the minimization of the number of passes of sweeping transducers. In particular, our minimization result for registers extends [4] to functional transducers with arbitrary output alphabets.

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 succint 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. Such 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 thus based on variants of the twinning property and are quite different from ours.

Overview. After introducing two-way and streaming transducers in Sections 2 and 3, and showing some basic properties, we recall in Section 4 the key characterization of one-way definability from [5]. Section 5 presents the main result on minimization of sweeping transducers. Finally, Section 6 concludes with a logical characterization for sweeping transducers.

2 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 $n \geq 2$, $a_1 = \triangleright$, $a_n = \triangleleft$, and $a_i \neq \triangleright, \triangleleft$ 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 $\triangleright$). 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. Indeed, if this were not the case, then there would exist a successful run visiting the same position twice with the same state and the same direction. By repeating the factor of the run between the two locations with the same state, one can generate other successful runs on the same input, thus contradicting unambiguity. It is convenient to assume that the number of passes performed by the successful runs of a transducer is uniformly bounded by a constant. In general, this is not possible unless the transducer is unambiguous. However, 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. This is without loss of generality, since for every successful run of a functional transducer, there is one that is successful, normalized, and that produces the same output.

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

We now turn to the subclass that we consider throughout the paper, namely *sweeping transducers*. Every k-pass R-sweeping transducer can be transformed into an equivalent $(k+1)$-pass L-sweeping transducer: the additional pass is used at the beginning of the run to move the head from the leftmost position to the rightmost position, without outputting anything. 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. As a matter of fact, this also implies that *bounded-reversal* transducers have the same expressiveness as sweeping 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.*

Proof. The first claim is trivial, so we focus on the remaining ones. We begin by considering an *unambiguous* $(k-1)$-reversal two-way transducer T, and we show how to transform it into an equivalent unambiguous k-pass sweeping transducer S. Towards the end of the proof we will discuss how to modify the constructions in order to transform an arbitrary $(k-1)$-reversal two-way transducer.

Hereafter, we refer to a *pass* of T as a maximal factor of a run of T with no reversals. Note that a pass of T might not start or end at the extremities of the input word. The goal is to simulate each pass of T with a pass of a suitable sweeping transducer S. In particular, when T performs a reversal, S needs to continue moving towards the extremity of the input (this will be either the leftmost or the rightmost position, depending on the parity of the

level), without performing any output. Once the extremity is reached, S is allowed to make a reversal and move back towards the position where T made the last reversal. So the transducer S moves from one extremity of the input to the other. The main difficulty is to correctly guess the reversals when moving away from these extremities. To do so, S will keep track, not only of the states associated with the current pass of T, but also of the crossing sequences of the entire successful run of T. If, at any point of the computation, a reversal position is wrongly guessed, this can be detected by the fact that it will be impossible to guess the crossing sequences across all the positions, since otherwise one could witness the existence of different successful runs on the same input. Note that the above idea works thanks to the fact that T is unambiguous: if this were not the case, then S may be able to guess different crossing sequences during different passes, and possibly produce a different output than T.

We will show later how to deal with the case where T is not unambiguous. Below, we provide more details about the construction that we just described.

For simplicity, we only consider the case where all successful runs of T start at the leftmost position and end at the rightmost position (the more general scenario can be handled by a case distinction and similar constructions). As T performs at most $k-1$ reversals, the crossing sequences of its successful runs will have length at most k.

To properly reason with crossing sequences, we need to introduce some additional terminology and notation. Let $u = a_1 \dots a_n$ be an input for T, where $a_1 = \triangleright$ and $a_n = \triangleleft$, and let ρ be a successful run of T on u. Recall that the crossing sequences $\rho|x$ are defined only at the positions $x \in \{1, \dots, n-1\}$. However, it is convenient to associate some crossing sequences also with the dummy positions 0 and n; we do so by letting $\rho|0 = \rho|n = \varepsilon$. Consider a pair of adjacent positions $x, x+1$, with $0 \leq x, x+1 < n$. Besides the crossing sequences $\rho|x$ and $\rho|x+1$, we can also identify the set of transitions of T that connect the locations at the positions x and $x+1$. We call these transitions the *crossing transitions of ρ between x and $x+1$*. Note that the crossing transitions between x and $x+1$ share the same input letter a_{x+1}. Moreover, if h_1 and h_2 are the lengths of the crossing sequences $\rho|x$ and $\rho|x+1$, respectively, then we see that there are exactly $\frac{h_1+h_2}{2}$ crossing transitions between x and $x+1$, and these can be ordered on the basis of the levels of their locations. We denote by $\rho|x, x+1$ the tuple of crossing transitions between x and $x+1$, ordered from bottom to top.

Let Q be the state space of T and E the set of its transition rules. For short, we denote by $Q^{\leq k}$ the set of tuples of distinct states in Q having length at most k, and we do similarly for $E^{\leq k}$. We introduce a relation $R \subseteq Q^{\leq k} \times E^{\leq k} \times Q^{\leq k}$ that represents the possible pairs of adjacent crossing sequences with the crossing transitions between them. We omit the tedious details of the definition of R and we refer the interested reader to [12]. What is important, here, is to know that one can define R so as to satisfy the following properties:

- if ρ is a successful run of T on $u = a_1 \dots a_n$, where $a_1 = \triangleright$ and $a_n = \triangleleft$, then R contains the triples $(\rho|x, \rho|x, x+1, \rho|x+1)$, for all $0 \leq x < x+1 < n$,
- if $u = a_1 \dots a_n$, with $a_1 = \triangleright$ and $a_n = \triangleleft$, and $(\overline{q}_0, \overline{e}_1, \overline{q}_1), \dots, (\overline{q}_{n-1}, \overline{e}_n, \overline{q}_n)$ is a sequence of triples in R such that, for each $1 \leq x < n$, the transitions in $\overline{e}_x$ have a_x as input symbol, then there is a successful run ρ of T on u such that $\rho|x = \overline{q}_x$ for all $0 \leq x \leq n$ and $\rho|x, x+1 = \overline{e}_x$ for all $1 \leq x \leq n$.

In particular, since T is unambiguous, for each input $u = a_1 \dots a_n$, there is at most one sequence $(\overline{q}_0, \overline{e}_1, \overline{q}_1), \dots, (\overline{q}_{n-1}, \overline{e}_n, \overline{q}_n) \in R^\star$ that satisfies the conditions of the second item.

We are now ready to construct the sweeping transducer S. Let us first define the notion of crossing index. We say that a location $\ell = (x, y)$ of a run ρ of a two-way transducer has *crossing index i* if ρ contains exactly i locations at position x and strictly below level y.

The state space of S is the set $Q^{\leq k} \times \{1, \dots, k\} \times \{\mathsf{follow}, \mathsf{escape}, \mathsf{reach}\}$. Intuitively, if a state $(\overline{q}, i, \mathsf{follow})$ marks a location $\ell = (x, y)$ of a successful run of S, then this means that there is a corresponding successful run ρ of T such that $\overline{q}$ is the crossing sequence $\rho|x$, ℓ is a location of ρ, and i is its crossing index of ℓ in ρ — in particular, this means that $\overline{q}[i]$ is the state associated with the location ℓ in the run ρ of T. The states of the form $(\overline{q}, i, \mathsf{escape})$ and $(\overline{q}, i, \mathsf{reach})$ are used for moving, respectively, away and towards the last reversal of T (so, intuitively, the locations marked with these states are "outside" the run of T).

As for the transition rules of S, below we describe those that are used to simulate a rightward pass of T (those simulating a leftward pass are just symmetric):

- For each triple $(\overline{q}, \overline{e}, \overline{q}') \in R$ and each transition $(\overline{q}[i], a, v, \overline{q}'[j], \mathsf{right}) \in \overline{e}$, we add to S the transition $\big((\overline{q}, i, \mathsf{follow}), a, v, (\overline{q}', j, \mathsf{follow}), \mathsf{right}\big)$.
- For each triple $(\overline{q}, \overline{e}, \overline{q}') \in R$ and each transition $(\overline{q}[i], a, v, \overline{q}[j], \mathsf{left}) \in \overline{e}$ (i.e. a reversal), we add to S either the transition $\big((\overline{q}, i, \mathsf{follow}), a, v, (\overline{q}', j, \mathsf{escape}), \mathsf{right}\big)$ or the transition $\big((\overline{q}, i, \mathsf{follow}), \lhd, v, (\overline{q}', j, \mathsf{follow}), \mathsf{left}\big)$, depending on whether $a \neq \lhd$ or $a = \lhd$.
- For each triple $(\overline{q}, \overline{e}, \overline{q}') \in R$ and each even index $j \in \{1, \dots, k\}$, we add to S either the transition $\big((\overline{q}, j, \mathsf{escape}), a, \varepsilon, (\overline{q}', j, \mathsf{escape}), \mathsf{right}\big)$ or the transition $\big((\overline{q}, j, \mathsf{escape}), \lhd, \varepsilon, (\overline{q}, j, \mathsf{reach}), \mathsf{left}\big)$, depending on whether $a \neq \lhd$ or $a = \lhd$, where a is the input symbol of the transitions of $\overline{e}$.
- For each triple $(\overline{q}, \overline{e}, \overline{q}') \in R$ and each even index $j \in \{1, \dots, k\}$, we add to S the transition $\big((\overline{q}', j, \mathsf{reach}), a, \varepsilon, (\overline{q}, j, \mathsf{reach}), \mathsf{left}\big)$, where a is the input symbol of the transitions of $\overline{e}$.
- For each triple $(\overline{q}, \overline{e}, \overline{q}') \in R$ and each transition $(\overline{q}[i], a, v, \overline{q}[j], \mathsf{left}) \in \overline{e}$, we add to S the transition $\big((\overline{q}', j, \mathsf{reach}), a, \varepsilon, (\overline{q}, j, \mathsf{follow}), \mathsf{left}\big)$.

The initial states of S are the triples $(\overline{q}', 1, \mathsf{follow})$, for any $\overline{q}'$ that appears in a triple of R of the form $(\varepsilon, \rhd, \overline{q}')$. Similarly, the final states of S the triples $(\overline{q}, i, \mathsf{follow})$, for any even index i and any $\overline{q}$ that appears in a triple of R of the form $(\overline{q}, \lhd, \varepsilon)$.

It is clear that the transducer S is k-pass sweeping and can be constructed in exponential time from T. Moreover, it is routine to verify that S can simulate any successful run of T. To prove that S is unambiguous, and thus equivalent to T, we consider the passes of S on some input u. Since each pass spans across all the positions of u, the series of crossing sequences guessed along the passes of S are always the same, and they represent a successful run of T on u. From this, using the fact that T is unambiguous, we easily verify that there exists at most one successful run of S on u. This proves that S is unambiguous and hence equivalent to T.

We explain now how to modify the construction of S when T is functional but not necessarily unambiguous. Consider an input word $u = a_1 \dots a_n$. The problem with T being not unambiguous is that S might guess different series of crossing sequences along different passes on the same input u, which can lead to producing an output different from that of T. The solution to this problem is to keep track of the set of possible crossing sequences $\rho|x$ that can be associated with a position x when ρ ranges over all successful runs of T on u. We call these crossing sequences the *admissible crossing sequences of u on x*, and we denote by C_x the set of all admissible crossing sequences of u on x.

Recall that R is the relation describing pairs of adjacent crossing sequences toghether with crossing transitions between them. Using R, we can derive the following relationship between the sets C_x and C_{x+1} associated with adjacent positions:

$$\begin{aligned} C_{x+1} &= \{\overline{q}' \,:\, \overline{q} \in C_x,\ (\overline{q}, \overline{e}, \overline{q}') \in R,\ a_{x+1} \text{ is the input symbol of the transitions in } \overline{e}\} \\ C_x &= \{\overline{q} \,:\, \overline{q}' \in C_x,\ (\overline{q}, \overline{e}, \overline{q}') \in R,\ a_{x+1} \text{ is the input symbol of the transitions in } \overline{e}\}. \end{aligned}$$

This relationship can be used by S in order to guess the sets C_x while moving across the positions of the input. In fact, there is a "warm-up" phase at the beginning of the computation, where there is no guarantee that S guesses the correct sets C_x. This phase lasts until the input u is entirely processed. During the warm-up phase, S may indeed guess over-approximations of the sets C_x, which are compatible with the possible continuations of the input. This is not problematic, because the only point where having an exact information about the sets C_x is crucial is when S guesses the position of the last reversal performed by T, which can only happen after the warm-up phase.

With the information derived from the guessed sets C_x, the transducer S can keep track of all the successful runs of T on u. It can then exploit non-determinism to guess and simulate a distinguished successful run, say the least one in some lexicographic order. The simulation carried out by S may fail in two cases: if the simulated run dies (as before, this can happen due to some wrong guesses), or if some other run that precedes the simulated one in the lexicographic order survives until the end (this means that the simulated run was not the least one). If S succeeds in simulating the distinguished run of T, then it can also guess the reversals of T along this run. We omit the formal definition of S, which is rather technical. We only remark that, as we need to consider sets of crossing sequences, here we incur into a doubly exponential blowup when transforming T into S. ◀

3 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 $R \uplus \Delta$ and to updates, as follows. For every valuation $g : R \to \Delta^\star$ and every word $w \in (R \uplus \Delta)^\star$, 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 $f : R \to (R \uplus \Delta)^\star$, 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

$$\sigma \;=\; (q_0, g_0) \xrightarrow[T]{a_1} (q_1, g_1) \xrightarrow[T]{a_2} \dots \xrightarrow[T]{a_n} (q_n, g_n).$$

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 copyless, (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, non-deterministic streaming transducers can be determinized. This happens at the cost of increasing the number of registers.

We show below an interesting correspondence between the number of registers in a natural subclass of streaming transducers and the number of passes performed by sweeping transducers.

▶ **Definition 2.** A streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$ is *concatenation-free* if it is copyless and $f(z) \in \Delta^\star \cdot (R \cup \{\varepsilon\}) \cdot \Delta^\star$, 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. Moreover, thanks to the previous Proposition 1, this correspondence can be immediately extended to the number of reversals performed by two-way transducers.

▶ **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* P *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.*

Proof. We begin by considering an *unambiguous* concatenation-free streaming transducer T with k registers, and we show how to transform it into an equivalent unambiguous $2k$-pass R-sweeping transducer S. After this, we will show how to deal with the case where the streaming transducer is not unambiguous.

Let $u = a_1 \dots a_n$ be an arbitrary input for T and $\sigma = (q_0, g_0) \xrightarrow[T]{a_1} (q_1, g_1) \xrightarrow[T]{a_2} \dots \xrightarrow[T]{a_n} (q_n, g_n)$ the unique successful run on u. We can write the corresponding output as

$$T(u) \;=\; v_0 \cdot g_n(z_1) \cdot v_1 \cdot g_n(z_2) \cdot \ldots \cdot g_n(z_h) \cdot v_h$$

where $h \leq k$ and $F(q_n) = v_0 \cdot z_1 \cdot v_1 \cdot z_2 \cdot \ldots \cdot z_h \cdot v_h$ (note that the latter word, and in particular the order of the registers $z_1, \dots, z_h$, depends on the final state q_n). Further let

$f_1, \ldots, f_n$ be the sequence of register updates induced by the above transitions, and recall that $g_x = g_{x-1} \circ f_x$ for all $x = 1, \ldots, n$.

The idea for obtaining an equivalent unambiguous $2k$-pass sweeping transducer S is to output each factor $v_i \cdot g_n(z_i)$ during two consecutive passes that start and end at the rightmost position. For this, we fix an index $i \in \{1, \ldots, h\}$ and we consider how the factor $g_n(z_i)$ is produced along the run σ. Since T is concatenation-free and unambiguous, there exist a unique position $x_i \in \{0, \ldots, n\}$, a unique sequence of registers $z_i = z_{i,n}$, $z_{i,n-1}$, …, $z_{i,x_i} \in R$, and a unique sequence of words $w_{i,n}, w'_{i,n}$, …, $w_{i,x_i}, w'_{i,x_i} \in \Delta^\star$ such that

$$\begin{cases} g_n(z_{i,n}) &= w_{i,n} \cdot g_{n-1}(z_{i,n-1}) \cdot w'_{i,n} \\ & \vdots \\ g_{x_i+1}(z_{i,x_i+1}) &= w_{i,x_i+1} \cdot g_{x_i}(z_{i,x_i}) \cdot w'_{i,x_i+1} \\ f_{x_i}(z_{i,x_i}) &= w_{i,x_i} \cdot w'_{i,x_i}. \end{cases}$$

This basically means that the factor $g_n(z_i)$ is obtained from the empty word by repeatedly prepending and appending finite words $w_{i,x}$ and $w'_{i,x}$, where x goes from x_i to n. Moreover, each pair of words $w_{i,x}$ and $w'_{i,x}$ is determined by the x-th transition $(q_{x-1}, g_{x-1}) \xrightarrow[T]{a_x} (q_x, g_x)$ of T.

The sweeping transducer S that we have to construct will behave as follows. At the beginning of a right-to-left pass at level $2i-1$, S outputs the word v_i that precedes the factor $g_n(z_n)$ in $T(u)$. During the same pass, it reads the input in backward direction and, while guessing the transitions $(q_{x-1}, g_{x-1}) \xrightarrow[T]{a_x} (q_x, g_x)$, it outputs the corresponding words $w_{i,x}$. Once the position x_i is reached, S continues to move leftward while simulating the transitions of T, but this time without producing any output. This is needed to check that the state of T reached at the leftmost position is initial. Then S performs a reversal. Similarly, during a left-to-right pass at level $2i$, the transducer S guesses the transitions $(q_{x-1}, g_{x-1}) \xrightarrow[T]{a_x} (q_x, g_x)$ and, if $x \geq x_i$, it outputs the corresponding words $w'_{i,x}$. Once the rightmost position is reached, it checks that the last guessed state is final and performs another reversal. The last pass performed by S is the left-to-right pass at level $2h$, where the last piece v_h of the output can be produced.

Clearly, S can be implemented with approximately $2k$ copies of the state space of T (one copy for each pass), and it performs at most $2k$ passes on any input. We remark that for this construction it is crucial that the streaming transducer T is unambiguous, as otherwise the described R-sweeping transducer S may guess different runs along two consecutive passes, eventually producing an output that differs from $T(u)$.

We know consider the situation where the streaming transducer T is not unambiguous. This case can be handled by preprocessing T so as to make it unambiguous. After this, one can apply the previous construction to obtain an equivalent $2k$-pass R-sweeping transducer. The transformation from a k-register streaming transducer T into an equivalent, unambiguous k-register streaming transducer T' can be performed in exponential time by using standard techniques. More precisely, one performs a subset construction on the finite state automaton A underlying T. This allows one to simulate deterministically all successful runs of A on the input word. Then one exploits non-determinism to canonically guess a single run among the successful ones — as usual, this can be the least run in some lexicographic ordering. Finally, one simulates the register updates of T along the guessed successful run. The resulting transducer T' has the same number of registers as T, but exponentially many more states. In particular, T' can be produced in exponential time from T.

For the converse translations, we start with a unambiguous k-pass R-sweeping transducer S and we show how to produce, in exponential time, an equivalent unambiguous concatenation-free streaming transducer S with $\lceil \frac{k}{2} \rceil$ registers.

Let A be the non-deterministic sweeping automaton underlying S, which recognizes the language $\mathsf{dom}(S)$. By applying the classical construction based on crossing sequences [12], we transform A into an equivalent unambiguous one-way automaton B. To obtain a streaming transducer equivalent to S, we equip the automaton B with $\lceil \frac{k}{2} \rceil$ registers, say $z_1, \ldots, z_{\lceil \frac{k}{2} \rceil}$, and we extend the transitions with suitable register updates. The idea is that each register z_i stores the output produced along the passes of S at levels $2i-1$ and $2i$. Consider an arbitrary input $u = a_1 \ldots a_n$ for S, where $a_1 = \rhd$ and $a_n = \lhd$, and recall that B admits a unique run on $a_2 \ldots a_{n-1}$, say

$$\sigma \;=\; s_1 \xrightarrow[B]{a_2} s_2 \xrightarrow[B]{a_3} \cdots \xrightarrow[B]{a_{n-1}} s_{n-1}.$$

Recall that the run σ determines a unique successful run ρ of S on u. In particular, each state s_x determines a crossing sequence $\rho|x$. If $\rho|x = (q_1, \ldots, q_h)$ and $\rho|x+1 = (q'_1, \ldots, q_h)$ are the crossing sequences of ρ at two adjacent positions x and $x+1$, for some $h \leq k$, then the corresponding update f_{x+1} for the registers of B must satisfy:

$$f_{x+1}(z_i) \;=\; w_{x+1} \cdot z_i \cdot w'_{x+1}$$

where w_{x+1} is the output produced by S with the right-to-left transition $q_{2i-1} \xleftarrow[S]{a_{x+1}} q'_{2i-1}$ and w'_{x+1} is the output produced by S with the left-to-right transition $q_{2i} \xrightarrow[S]{a_{x+1}} q'_{2i}$. Since f_{x+1} is uniquely determined by the control states s_x and s_{x+1} and the input symbol a_{x+1}, the above equations can be easily turned into a definition of transition rules for a streaming transducer T having B as underlying automaton. We then specify the partial output function of T, which maps any state s of B to the juxtaposition $z_1 \cdot \ldots \cdot z_{\lceil \frac{h}{2} \rceil}$ of the first $\lceil \frac{h}{2} \rceil$ registers, where h is the length of the crossing sequence determined by s. The resulting transducer T is unambiguous, uses at most $\lceil \frac{k}{2} \rceil$ registers, and is equivalent to S.

To conclude the proof, we briefly discuss the case where the sweeping transducer S is not unambiguous. As usual, this case can be deal with by a form of subset construction. The key observation is that all successful runs of S can be followed simultaneously because they perform the same reversals at the same positions, namely, at the extremities of the input word. In other words, one can determinize the underlying sweeping automaton in simple exponential time. ◀

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 5, 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.

4 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 $\mathsf{pump}^n_L(u) = u[1, x_1] \cdot \big(\boldsymbol{u[x_1 + 1, x_2]}\big)^{\boldsymbol{n}} \cdot u[x_2 + 1, |u|]$ and we observe that $\mathsf{pump}^n_L(\rho)$ is a valid run on $\mathsf{pump}^n_L(u)$. We remark that the above definition of pumped run are correct only for sweeping transducers: for arbitrary two-way transducers we would need to take into account the possible reversals within a loop L and combine the intercepted factors in a more complex way.

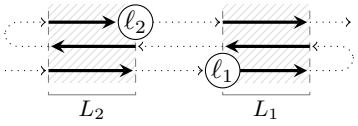
It is convenient to introduce some notation for pumping runs on multiple loops. If the loops are pairwise non-overlapping this can be done by simply pumping each loop separately, since the order in which we pump the loops does not really matter. The situation is a bit more complicated when some loops 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. A crucial result from [5] shows that, if S is equivalent to a one-way transducer, then all outputs $\mathsf{out}(\rho[\ell_1, \ell_2])$ produced between the locations of an inversion of ρ are periodic, and the periods are uniformly bounded by the constant e_S. This result is obtained by considering the output produced by pumped runs of the form $\mathsf{pump}_{L_2}^{n_2}(\mathsf{pump}_{L_1}^{n_1})(\rho))$ and between copies of the locations ℓ_1 and ℓ_2. By observing how these outputs are covered by the words produced by an equivalent one-way transducer T, and by exploiting Fine-Wilf's Theorem, one derives the periodicity of the outputs.

▶ **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.*

Based on the above result, it makes sense to define the language $L_S \subseteq \mathsf{dom}(S)$ of all input 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 then characterized as follows:

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

In the next section we show how to generalize the above results in order to characterize k-pass definability.

5 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. As usual, 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 that of Definition 4 (specifically, it is obtained by reversing the order of the witnessing loops):

▶ **Definition 7.** A *co-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:

- ℓ_1 is an entry point of L_1 and ℓ_2 is an exit point of L_2,
- $\ell_1 < \ell_2$ and $\boldsymbol{x_1 \leq x_2'}$,
- 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.

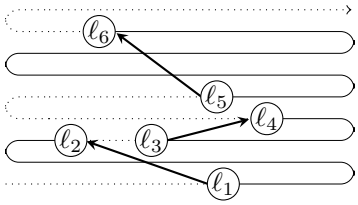
We say that the above loops L_1 and L_2 are the *witnessing loops* of the co-inversion (ℓ_1, ℓ_2).

We then combine inversions and co-inversions, as follows:

▶ **Definition 8.** 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 on 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 $u \in \mathsf{dom}(S)$ such that all k-inversions of all successful runs of S on u are safe.



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 $L_S^{(1)} = \mathsf{dom}(S)$. The generalization of this result is 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 $L_S^{(k)} = \mathsf{dom}(S)$, 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 $L_S^{(k)} = \mathsf{dom}(S)$ 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 $L_S^{(k)} = \mathsf{dom}(S)$ 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.

The following lemma shows that we can equally reason in terms of safe k-inversions (Definition 8) or k-factorizations.

▶ **Lemma 12.** *For every word $u \in \mathsf{dom}(S)$, we have that $u \in L_S^{(k)}$ if and only if all successful runs of S on u admit k-factorizations.*

Proof. We prove the left-to-right direction. Let $u \in L_S^{(k)}$ and let ρ be a successful run of S on u. We define the locations $\ell_0, \ell_1, \ldots, \ell_k$ forming a k-factorization of ρ by an iterative process. The location ℓ_0 is clearly the first of the run. Let us now assume that we have defined the locations up to ℓ_i, with $i < k$. We distinguish two cases depending on the parity of i. If i is odd, then we look at the inversions (ℓ, ℓ') of ρ such that $\ell \geq \ell_i$ and $\mathsf{out}(\rho[\ell, \ell'])$ has period strictly larger than e_S. For short, we call such inversions *bad inversions after* ℓ_i. If there are no bad inversions after ℓ_i, then we simply define ℓ_{i+1} to be the last location of the run. Otherwise, we take the first bad inversion after ℓ_i, following the lexicographic order on pairs of locations, and we denote it by $(\tilde{\ell}_i, \tilde{\ell}'_i)$. Accordingly, we define the next location ℓ_{i+1} to be the one that immediately precedes $\tilde{\ell}'_i$ in the run ρ. The case where i is even is dealt with in a similar way, by considering co-inversions instead of inversions.

We verify that the constructed sequence $\ell_0, \ell_1, \ldots, \ell_k$ is indeed a k-factorization. By definition, for all even (resp. odd) indexes $0 \leq i < k$ and all inversions (resp. co-inversions) (ℓ, ℓ'), with $\ell_i \leq \ell \leq \ell' \leq \ell_{i+1}$, the period of $\mathsf{out}(\rho[\ell, \ell'])$ is at most e_S. Moreover, ℓ_0 was chosen to be the first location of ρ, and we clearly have $\ell_0 \leq \ell_1 \leq \ldots \leq \ell_k$. It remains to prove that ℓ_k is the last location of ρ. Suppose that this is not the case. Of course, this can happen only if for each location ℓ_i, with $i = 1, \ldots, k$, there exist bad inversions (or co-inversions, depending on the parity of i) after ℓ_{i-1}. For each $i = 1, \ldots, k$, let $(\tilde{\ell}_i, \tilde{\ell}'_i)$ be the first bad (co-)inversion after ℓ_{i-1}. The sequence $(\tilde{\ell}_1, \tilde{\ell}'_1), \ldots, (\tilde{\ell}_k, \tilde{\ell}'_k)$ is clearly a k-inversion. Moreover, it is unsafe, because the period of every word $\mathsf{out}(\rho[\tilde{\ell}_i, \tilde{\ell}'_i])$ exceeds e_S. However, this is against the hypothesis that ρ was a successful run on $u \in L_S^{(k)}$. We thus conclude that $\overline{\ell} = \ell_0, \ell_1, \ldots, \ell_k$ is a k-factorization of ρ.

We now prove the converse direction. Fix a word u such that all successful runs on it admit k-factorizations. Consider a successful run ρ 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. As ρ is a successful run on u, it admits a k-factorization, say $\overline{\ell} = \ell_0, \ldots, \ell_k$. As the locations of the k-inversion are ordered, i.e. $\ell'_1 \leq \ell'_2 \leq \ldots \leq \ell'_{2k-1} \leq \ell'_{2k}$, there is an index $i \in \{0, \ldots, k-1\}$ such that $\ell_i \leq \ell'_{2i+1} \leq \ell'_{2i+2} \leq \ell_{i+1}$. By definition of k-factorization, this means that the period of $\mathsf{out}(\rho[\ell'_{2i+1}, \ell'_{2i+2}])$ is at most e_S, and hence the k-inversion $\overline{\ell}'$ is safe. As we chose ρ and $\overline{\ell}'$ in a general way, we conclude that $u \in L_S^{(k)}$. ◀

Now, 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 $u \in \mathsf{dom}(S)$, 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 13 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 13.** *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.*

Proof. To prove that $F_S^{(k)}$ is recognized by an automaton of doubly exponential size, we rely again on results from [5]. Recall that L_S is the language of words that induce successful runs such that, for all inversions (ℓ, ℓ'), the period of $\mathsf{out}(\rho[\ell, \ell'])$ is at most e_S. Theorem 6 shows that L_S is the domain of a one-way transducer T that can be constructed from S in doubly exponential time, so L_S is recognized by an automaton of doubly exponential size w.r.t. S. We can apply this theorem to the transducer S_{factors}, and obtain in this way an automaton that recognizes the language $L_{S_{\text{factors}}}$. By definition, if ρ is a successful run on u and ℓ_1, ℓ_2 are two locations in it, then $\langle u, \rho, \ell_1, \ell_2\rangle$ belongs to $L_{S_{\text{factors}}}$ if and only if, for all inversions (ℓ, ℓ'), with $\ell_1 \leq \ell \leq \ell' \leq \ell_2$, the period of $\mathsf{out}(\rho[\ell, \ell'])$ is at most e_S. A similar automaton can be constructed for the language $R_{S_{\text{factors}}}$ that contains the words $\langle u, \rho, \ell_1, \ell_2\rangle$ such that the outputs produced by all *co*-inversions between ℓ_1 and ℓ_2 have period at most e_S. Therefore, to recognize $F_S^{(k)}$, it is sufficient to construct an automaton that reads a word $\langle u, \rho, \ell_0, \ldots, \ell_k\rangle$ and checks that (i) ρ is a successful run of S on u, (ii) $\ell_0 \leq \ldots \leq \ell_k$ are locations that delimit factors of ρ, and (iii) for every $i \in \{0, \ldots, k-1\}$, $\langle u, \rho, \ell_i, \ell_{i+1}\rangle$ belongs to either $L_{S_{\text{factors}}}$ or $R_{S_{\text{factors}}}$, depending on the parity of i. A close inspection to the above constructions shows that all the automata can be produced in doubly exponential time w.r.t. S.

We now show that the complement $\overline{F}_S^{(k)}$ can also be recognized by an automaton of doubly exponential size. Indeed, checking that a word $\langle u, \rho, \overline{\ell}\rangle$ does *not* belong to $F_S^{(k)}$ boils down to verifying that one the following conditions holds:

1. ρ is not a successful run on u. This can be checked by looking at the crossing sequences annotated on the positions of u, using an exponential number of states.
2. The locations in $\overline{\ell}$ do not define a factorization of ρ. This can be easily checked with polynomially many states.
3. There exist an index $i \in \{0, \ldots, k-1\}$ and an inversion (ℓ, ℓ') (or a co-inversion, depending on the parity of i), with $\ell_i \leq \ell \leq \ell' \leq \ell_{i+1}$, such that, for all periods $0 \leq p \leq e_S$, $\mathsf{out}(\rho[\ell, \ell'])[z] \neq \mathsf{out}(\rho[\ell, \ell'])[z + p]$ for some position z — note that this is equivalent to saying that $\overline{\ell}$ is not a k-factorization. The latter condition can be checked by guessing i, (ℓ, ℓ'), and a function that maps numbers $p \in \{0, \ldots, e_S\}$ to positions z_p in $\mathsf{out}(\rho[\ell, \ell'])$. This can be done with doubly exponentially many states. ◀

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 13. 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 13, 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 $L_S^{(k)} = \mathsf{dom}(S)$. In fact, as we already know that $L_S^{(k)} \subseteq \mathsf{dom}(S)$, it suffices to decide only the containment $L_S^{(k)} \supseteq \mathsf{dom}(S)$. We know from Lemma 12 that the language $L_S^{(k)}$ coincides with the projection of $F_S^{(k)}$ on the underlying words u. Thus, we have

$$L_S^{(k)} \supseteq \mathsf{dom}(S) \qquad \text{if and only if} \qquad \overline{F}_S^{(k)} \cap D = \emptyset$$

where $D = \{\langle u, \rho, \overline{\ell} \rangle \ : \ u \in \mathsf{dom}(S)\}$. 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

$$\rho^{\overline{n}} \;=\; \mathsf{pump}^{\overline{n}}_{\overline{L}}(\rho)$$

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 4). 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 $i \in \{1, \ldots, 2k\}$ (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 $i = 0, \ldots, k-1$, we define

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

where $(\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2})$ is the (co-)inversion of $\rho^{\overline{n}}$ that corresponds to $(\ell_{2i+1}, \ell_{2i+2})$. Note that the above factors depend on the k-inversion $\overline{\ell}$ and on the tuple $\overline{n}$, which represents the number of times we pump each witnessing loop of $\overline{\ell}$. For simplicity, since $\overline{\ell}$ is understood from the context, we will often drop the subscript from the notation, thus writing $v^{\overline{n}}(i)$. 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 $j = 0, \ldots, k-1$, 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 14.** 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 $\{1, \ldots, 2k\}$. 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 $i = 0, \ldots, k-1$:

$$\mathcal{P}_i(\overline{n}) \;=\; \text{“}v^{\overline{n}}(i) \textit{ is a factor of } w^{\overline{n}}(i)\text{”}.$$

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 15.** *There is 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* $\overline{n}$.

Proof. The proof is rather technical, and we need to reason on coverability between pairs of factors $v^{\overline{n}}(i)$ and $w^{\overline{n}}(j)$, for possibly distinct indexes $i, j \in \{0, \ldots, k-1\}$. For the sake of brevity, we denote by $C_{i,j}$ the set of tuples $\overline{n} \in \mathbb{N}^{2k}$ such that $v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(j)$. Similarly, we denote by $D_{i,j}$ the set of tuples $\overline{n} \in \mathbb{N}^{2k}$ such that the suffix $\mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_{2i+1}, \tilde{\ell}_{2k+1}]\big)$ of $S(u^{\overline{n}})$ — i.e. the one that starts with $v^{\overline{n}}(i)$ — is a factor of the suffix $\mathsf{out}\big(\sigma^{\overline{n}}[\ell'_j, \ell'_k]\big)$ of $T(u^{\overline{n}})$ — i.e. the one that starts with $w^{\overline{n}}(j)$. Note that $D_{0,0} = \mathbb{N}^{2k}$ since $S(u^{\overline{n}}) = T(u^{\overline{n}})$. For convenience, we also let $C_{i,j} = D_{i,j} = \emptyset$ whenever $i = k$ or $j = k$.

▶ **Claim.** *For all* $i, j \in \{0, \ldots, k-1\}$, *we have* $D_{i,j} \subseteq C_{i,j} \cup D_{i+1,j+1}$.

Proof. Consider a tuple $\overline{n}$ in $D_{i,j}$. By definition, we know that the suffix of $S(u^{\overline{n}})$ that starts with $v^{\overline{n}}(i)$ is a factor of the suffix of $T(u^{\overline{n}})$ that starts with $w^{\overline{n}}(j)$. We distinguish some cases depending on whether $j = k-1$ or $j < k-1$, and whether $\overline{n} \in C_{i,j}$ or not. If $j = k-1$, we prove the claim by observing that $\overline{n}$ must belong to $C_{i,j}$: indeed, if this is not the case, then the last factor $w^{\overline{n}}(k-1)$ of $T(u^{\overline{n}})$ would end strictly before the end of the factor $v^{\overline{n}}(i)$, thus contradicting the hypothesis that $S(u^{\overline{n}}) = T(u^{\overline{n}})$. If $j < k-1$ and $\overline{n} \in C_{i,j}$, then the claim holds trivially. Finally, suppose that $j < k-1$ and $\overline{n} \notin C_{i,j}$. Since $\overline{n} \in D_{i,j}$, we know that the factor $v^{\overline{n}}(i)$ begins after the beginning of $w^{\overline{n}}(j)$. However, because $v^{\overline{n}}(i)$ is not a factor of $w^{\overline{n}}(j)$, we also know that $v^{\overline{n}}(i)$ ends after the ending of $w^{\overline{n}}(j)$. This implies that $v^{\overline{n}}(i+1)$ begins after the beginning of $w^{\overline{n}}(j+1)$, whence $\overline{n} \in D_{i+1,j+1}$. ◀

Using the above claim we can derive the first important equation:

$$\mathbb{N}^{2k} \;=\; D_{0,0} \;\subseteq\; C_{0,0} \cup D_{1,1} \;\subseteq\; \ldots \;\subseteq\; \bigcup\nolimits_i C_{i,i} \;\cup\; D_{k,k} \;=\; \bigcup\nolimits_i C_{i,i}. \tag{3}$$

This basically means that it is sufficient to consider only the coverability between pairs of factors $v^{\overline{n}}(i)$ and $w^{\overline{n}}(i)$, having the same index i.

Now, recall that $\mathcal{P}_i(\overline{n})$ denotes the property “$v^{\overline{n}}(i)$ *is a factor of* $w^{\overline{n}}(i)$ ”. The latter property can be equally stated as $\overline{n} \in C_{i,i}$, and thus can be seen as a special case of a more general property, parametrized by subsets of $\mathbb{N}^{2k}$. Formally, for every set $N \subseteq \mathbb{N}^{2k}$, we define the property

$$\mathcal{R}_N(\overline{n}) \;=\; \text{“}\,\overline{n} \in N\,\text{”}.$$

Clearly, the statement of Lemma 15 is equivalent to saying that there exist an index $i \in \{0,\dots,k-1\}$ and a set $N \subseteq \mathbb{N}^{2k}$ such that $\mathcal{R}_{N\cap C_{i,i}}(\overline{n})$ holds unboundedly on the coordinates $2i+1$ and $2i+2$. Based on this, we can prove Lemma 15 by way of contradiction, namely, by assuming that for all $i = 0,\dots,k-1$ and all $N \subseteq \mathbb{N}^{2k}$, $\mathcal{R}_{N\cap C_{i,i}}$ does *not* hold unboundedly on the coordinates $2i+1$ and $2i+2$. More precisely, we will use the latter assumption and exploit an induction on $i = 0,\dots,k$ to construct some sets $N_i \subseteq \mathbb{N}^{2k}$ satisfying the following conditions:

1. N_i is a Cartesian product of the form $\{\overline{m}_i\} \times \mathbb{N}^{2k-2i}$, for some tuple $\overline{m}_i \in \mathbb{N}^{2i}$,
2. if $i > 0$, then $N_i \subseteq N_{i-1}$,
3. $N_i \subseteq \bigcup_{j\geq i} C_{j,j}$.

We will reach a contradiction when $i = k$, since the first condition implies that N_k is a singleton, while the last condition requires N_k to be empty (indeed, $\bigcup_{j\geq k} C_{j,j}$ is the empty union).

For the base case $i = 0$, we simply let $N_0 = \mathbb{N}^{2k}$ and observe that conditions 1.–3. hold trivially (in particular, the third condition follows from Equation (3)).

As for the inductive step, we assume that $i < k$ and that N_i satisfies conditions 1.–3, and we show how to construct N_{i+1} satisfying analogous properties. The idea is to fix the components $2i+1$ and $2i+2$ in N_i in such a way that the resulting set is contained in $N_i \setminus C_{i,i}$. Recall that we assumed, towards a contradiction, that $\mathcal{R}_{N_i\cap C_{i,i}}(\overline{n})$ does not hold unboundedly on the coordinates $2i+1$ and $2i+2$. By Definition 14, this means that there exists a number $n_0 \in \mathbb{N}$ that satisfies one of the two cases below:

a) for all $\overline{n} \in N_i \cap C_{i,i}$, $\overline{n}[2i+1] < n_0$ or $\overline{n}[2i+2] - \overline{n}[2i+1] < n_0$,
b) for all $\overline{n} \in N_i \cap C_{i,i}$, $\overline{n}[2i+2] < n_0$ or $\overline{n}[2i+1] - \overline{n}[2i+2] < n_0$.

We define N_{i+1} on the basis of n_0 and of which case holds:

$$N_{i+1} \;=\; \begin{cases} \{\overline{n} \in N_i \;:\; \overline{n}[2i+1] = n_0,\ \overline{n}[2i+2] = 2n_0\} & \text{if case a) holds} \\ \{\overline{n} \in N_i \;:\; \overline{n}[2i+2] = n_0,\ \overline{n}[2i+1] = 2n_0\} & \text{if case b) holds.} \end{cases}$$

We now verify that N_{i+1} satisfies the conditions 1.–3. for the index $i+1$. The first condition holds by construction and thanks to the inductive hypothesis: indeed, N_i projected on the components $h \leq 2i$ is a singleton, and N_i projected on the components $h \geq 2i+1$ is $\mathbb{N}^{2k-2i}$. The second condition $N_{i+1} \subseteq N_i$ holds trivially by construction. As for the third condition, we first claim that $N_{i+1} \cap C_{i,i}$ is empty. Indeed, if N_{i+1} was defined from case a), then all the tuples $\overline{n}$ in it satisfy $\overline{n}[2i+1] = \overline{n}[2i+2] - \overline{n}[2i+1] = n_0$. If any of these tuples belonged also to $C_{i,i}$, we would get a contradiction with case a). The argument for case b) is just symmetric. Now that we know that $N_{i+1} \cap C_{i,i} = \emptyset$, we can easily derive $N_{i+1} \subseteq N_i \setminus C_{i,i} \subseteq \bigcup_{j\geq i+1} C_{j,j}$.

Summing up, we assumed, by way of contradiction, that for all $i = 0,\dots,k-1$ and all $N \subseteq \mathbb{N}^{2k}$, $\mathcal{R}_{N\cap C_{i,i}}(\overline{n})$ does not hold unboundedly on the coordinates $2i+1$ and $2i+2$. Using this assumption, we defined inductively some sets $N_0, N_1, \dots, N_k$ satisfying conditions 1.–3. We also observed that, for $i = k$, the conditions 1. and 3. contradict each other. From this we must conclude that, for some index i and some $N \subseteq \mathbb{N}^{2k}$, the property $\mathcal{R}_{N\cap C_{i,i}}(\overline{n})$ (and hence $\mathcal{P}_i(\overline{n})$ as well) holds unboundedly on the coordinates $2i+1$ and $2i+2$. ◀

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 15. 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 16.** *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.*

Proof. We begin by distinguishing two cases, depending on whether i is even or odd. If i is even, then $(\ell_{2i+1}, \ell_{2i+2})$ is an inversion and the factor $w^{\overline{n}}(i)$ is produced along a left-to-right pass of T, therefore like in a one-way transduction. Otherwise, if i is odd, then $(\ell_{2i+1}, \ell_{2i+2})$ is a co-inversion and the factor $w^{\overline{n}}(i)$ is produced along a right-to-left pass of T. As the two cases are symmetric, we can focus only on one of the two, say the case where i is even. The proof that $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ is periodic is exactly the same as that of Proposition 5. We briefly recall the crucial arguments below.

In that proof we considered an inversion (ℓ_1, ℓ_2) and the outputs produced by runs of the form $\mathsf{pump}_{L_2}^{n_2}(\mathsf{pump}_{L_1}^{n_1}(\rho))$ between the locations ℓ_1 and ℓ_2. The latter outputs were then compared with the outputs produced by an equivalent one-way transducer T. In particular, we observed that the former outputs are factors of the latter outputs. More precisely, by letting the parameters n_1 and n_2 grow independently, it was possible to exploit Fine-Wilf's Theorem and derive the periodicity of the former outputs.

The same argument can be applied here with the inversion $(\ell_{2i+1}, \ell_{2i+2})$ and the factors $v^{\overline{n}}(i)$ and $w^{\overline{n}}(i)$, produced respectively by S and T. Indeed, to apply Fine-Wilf's Theorem, it is sufficient that the coverability relationship holds for pairs of arbitrarily large numbers $\overline{n}[2i+1]$ and $\overline{n}[2i+2]$, and that these numbers can vary independently of each other. This is precisely what it means for the property $\mathcal{P}_i(\overline{n})$ to hold unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$. ◀

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 15, 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 16, we derive that the k-inversion $\overline{\ell}$ is safe. This proves the left-to-right direction of the first claim of Theorem 9. ◀

6 Sweeping transducers and MSO

Here, we give 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 $\Phi_a^i(x)$, one for each $i \in \{1, \ldots, m\}$ and $a \in \Delta$, and some binary MSO formulas $\Phi_{\leq}^{i,j}(x, y)$, 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 $\Phi_a^i(x)$ 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 $\Phi_{\leq}^{i,j}(x, y)$ 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 $u \models \Phi_{\mathsf{dom}}$. Similarly, we assume that $\Phi^{i,j}_{\leq}(x,y)$ entails $\Phi^i(x)$ and $\Phi^j(y)$.

▶ **Definition 17.** 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. Symmetrically, order-inversing MSO transductions capture the transductions definable by 1-pass R-sweeping transducers. So it is not surprising that k-phase MSO transductions correspond to k-pass L-sweeping transducers:

▶ **Theorem 18.** *k-phase MSO transductions have the same expressive power as functional, k-pass L-sweeping transducers.*

Proof. We first show how to translate a k-pass L-sweeping, functional transducer S into an equivalent k-phase MSO transduction. The technique is a variant of the classical translation from two-way transducers to MSO transductions (see for instance [9]).

First, thanks to Proposition 1, we can assume, without loss of generality, that the sweeping transducer S is unambiguous. Recall that S performs at most k passes, and let c_S be the maximal number of characters output by a single transition of S. To logically define the output of the MSO transduction, we will take $k \cdot c_S$ copies of the input. Each copy of the input is thus indexed by a pair (h,i), where $0 \leq h < k$ and $1 \leq i \leq c_S$. Intuitively, the copy indexed by (h,i) represents the i-th letter of the output produced during the pass at level h. Using the classical correspondence between automata and MSO, we can build an MSO sentence Φ_{dom} that tells whether a word u belongs to the domain of the transduction. Similarly, for each index $0 \leq h < k$ and each transition rule τ of S, we can build an MSO formula $\Phi_{h,\tau}(x)$ such that, for all $u \in \mathsf{dom}(S)$, $u \models \Phi_{h,\tau}(x)$ iff the transition rule τ is applied at the location $\ell = (x,h)$ of the unique successful run of S induced by u. We complete the definition of the MSO transduction equivalent to S as follows:

- For the formulas defining the output elements and their labels, we let $\Phi^{(h,i)}_a(x)$ be the disjunction of the formulas $\Phi_{h,\tau}(x)$, for all $0 \leq h < k$ and $\tau = (p,a,v,q)$ such that $|v| > i$ and $v(i) = a$. Note that, because only one transition rule can be used at each location of the successful run, for each $1 \leq x \leq |u|$ and each $0 \leq h < k$, there can exist at most letter a such that $u \models \Phi^{(h,i)}_a(x)$.
- For the formulas ordering the output elements, we define $\Phi^{(h,i),(h',i')}_{\leq}(x,y)$ by a case distinction:

$$\Phi^{(h,i),(h',i')}_{\leq}(x,y) \;=\; \begin{cases} \mathit{true} & \text{if } h < h' \\ x \leq y & \text{if } h = h' \text{ is even and } i \leq i' \\ x < y & \text{if } h = h' \text{ is even and } i > i' \\ x \geq y & \text{if } h = h' \text{ is odd and } i \leq i' \\ x > y & \text{if } h = h' \text{ is odd and } i > i'. \end{cases}$$

  It is easy to see that the formulas $\Phi^{(h,i),(h',i')}_{\leq}(x,y)$ define a total order on the output structure.

It remains to show that the above MSO transduction is k-phase. For this, we order lexicographically the index set $\{(h,i) \;:\; 0 \leq h < k,\ 1 \leq i \leq c_S\}$, and partition it into k subsets

$I_0 < I_1 < \ldots, I_{k-1}$, where $I_h = \{(h,i) \ : \ 1 \leq i \leq c_S\}$ for all $h = 0, 1, \ldots, k-1$. We then observe that the defined partition satisfies Definition 17: indeed, for all pairs $(h,i), (h,j) \in I_h$, $\Phi_{\leq}^{(h,i)(h,j)}(x,y)$ entails either $x \leq y$ or $x \geq y$, depending on whether h is even or odd.

We now prove the converse translation, from a k-phase MSO transduction to an equivalent k-pass L-sweeping transducer. Let Φ_{dom}, $\Phi_a^i(x)$, and $\Phi_{\leq}^{i,j}(x,y)$ be the formulas defining an arbitrary k-phase MSO transduction, where i, j range over some finite set I. Further let $I_0 < I_1 < \ldots, I_{k-1}$ be a partition of I satisfying the third item of Definition 17. Note that, for a fixed $0 \leq h < k$, the family of formulas $\Phi_a^i(x)$ and $\Phi_{\leq}^{i,j}(x,y)$ where i, j range over I_h define an MSO transduction that is either order-preserving or order-inversing, depending on whether h is even or odd. For the sake of brevity, we denote by T the original k-pass MSO transduction, and by $T_0, T_1, \ldots, T_{k-1}$ the corresponding order-preserving/order-inversing MSO transductions.

If each T_h maps an input word u to an output v_h, then we know that T maps u to the juxtaposition $v_0 \cdot v_1 \cdot \ldots \cdot v_{k-1}$. Moreover, we know from [9] that we can translate the order-preserving transductions $T_0, T_2, \ldots$ to equivalent one-way transducers $S_0, S_2, \ldots$. Similarly, we can translate the order-inversing transductions $T_1, T_3, \ldots$ to equivalent 1-pass R-sweeping transducers $S_1, S_3, \ldots$. Thus, we can obtain the desired k-pass L-sweeping transducer S by simply concatenating the transducers $S_0, S_1, S_2, S_3, \ldots$. ◀

7 Conclusions

We gave an effective characterization of transductions definable by sweeping transducers with k-passes. From the correspondence between the number of passes of a sweeping transducer and the number of registers required by an equivalent concatenation-free streaming transducer, we derived a procedure that minimizes the number of registers in a concatenation-free sweeping transducer. We also showed that sweeping transducers, bounded-reversal transducers, and concatenation-free streaming transducers define the same class of transductions. Finally, we provided a logical characterization of the latter class based on a restriction of MSO-definable transductions.

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. Available at `hal.archives-ouvertes.fr/hal-01201704`, 2015.

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.