

Finitary Compositions of Two-way Finite-State Transductions

Joost Engelfriet, Hendrik Jan Hoogeboom

LIACS, Leiden University

PO Box 9512, 2300 RA Leiden, The Netherlands

`{engelfri, hoogeboo}@liacs.nl`

Abstract. The hierarchy of arbitrary compositions of two-way nondeterministic finite-state transductions collapses when restricted to finitary transductions, i.e., transductions that produce a finite set of outputs for each input. The hierarchy collapses to the class of nondeterministic MSO definable transductions, which is inside the second level of that hierarchy. It is decidable whether a composition of two-way nondeterministic finite-state transducers realizes a finitary transduction (i.e., is MSO definable).

Introduction

Monadic second-order (MSO) logic can be used in a natural way to define graph transductions, see [4, 5]. To include nondeterminism into such a definition, the MSO formulas that constitute the definition may contain parameters: for each value of the parameters the input graph is translated into an output graph (but for some values of the parameters the output may be undefined). Since the parameters can only assume a finite number of values (viz. sets of nodes of the input graph), an MSO definable graph transduction is finitary, which means that for each input graph the set of output graphs is finite. In the parameterless, i.e., deterministic, case a function is defined.

Strings and trees are special cases of graphs, and both string transducers and tree transducers have been extensively investigated. Thus the question arises which string and tree transductions are MSO definable, in terms of well-known types of string and tree transducers. This question was investigated for strings in [9] and for trees in [3, 10, 11, 16].

It was shown in [9] that the deterministic MSO definable string transductions are exactly the transductions realized by two-way deterministic finite-state transducers, more commonly called 2DGSM's

(two-way deterministic generalized sequential machines). These are finite-state transducers with a two-way input tape (surrounded by endmarkers) and a one-way output tape. In the nondeterministic case the situation is not as simple as that. Three things were shown in [9]. First, the nondeterministic MSO definable string transductions are incomparable with the transductions realized by two-way nondeterministic finite-state transducers (2NGSM's), one reason being that 2NGSM's can compute nonfinitary transductions. Second, they can be computed by the composition of *two* 2NGSM's. And third, they are, in fact, precisely the transductions that can be realized by the composition of two so-called *finite-visit* 2NGSM's, where a 2NGSM is finite-visit if there is a bound k such that every output string can be computed from the input string by a computation that visits each cell of the input tape at most k times. This is a computational property of 2NGSM's.

In this paper we characterize the nondeterministic MSO definable string transductions by a behavioural property of compositions of 2NGSM's: we prove that they are exactly the finitary transductions that can be realized by the composition of two 2NGSM's. We extend this to arbitrary compositions of 2NGSM's: every finitary transduction that can be realized by an arbitrary composition of 2NGSM's, is MSO definable. Moreover, we show that it is decidable for an arbitrary sequence of 2NGSM's whether or not their composition realizes a finitary transduction, i.e., whether that transduction is MSO definable.

It should be noted that, as shown in [12, 7, 8], the $(n + 1)$ -fold compositions of 2NGSM's are more powerful than the n -fold compositions of 2NGSM's. Thus, our extended result shows that this hierarchy collapses for finitary transductions to the second level, more precisely, to the class of nondeterministic MSO definable string transductions. It is already known that the hierarchy collapses for functional transductions, to the class of transductions realized by 2DGSM's (see [7]), i.e., to the class of deterministic MSO definable string transductions.

These results for the nondeterministic MSO definable string transductions are analogous to those that are proved for the *deterministic* MSO definable *tree* transductions in [3, 10, 11, 16]. In fact, in [3, 10] it was shown that the latter transductions are exactly those realized by so-called finite-copying deterministic macro tree transducers, a result similar to the one in [9] using a computational property of a well-known class of transducers. Then, in [11] it was shown that they are exactly the LSI transductions that can be realized by a deterministic macro tree transducer, where a transduction is LSI if the size of the output is linear in the size of the input: a behavioural property of transducers. In that same paper it is shown that it is decidable for a deterministic macro tree transducer whether it realizes an LSI transduction, i.e., whether that transduction is MSO definable. In [16] this was extended to arbitrary compositions of deterministic macro tree transducers. It is open whether similar results hold for *nondeterministic* MSO definable *tree* transductions.

In this paper we do not discuss logic. Instead, we rely on the characterizations of the MSO definable string transductions in terms of 2NGSM's, as proved in [9].

1. Two-way transducers

A *linear bounded transducer* is a finite-state device equipped with a two-way input tape and a one-way output tape. The input string is initially placed on the input tape, surrounded by the endmarkers \vdash and \dashv , that indicate the left and right boundary of the input tape, respectively. The device can read and rewrite the symbols on the input tape, using its reading head just like a Turing machine or, more precisely, a linear bounded automaton (because it cannot move beyond the endmarkers).

A linear bounded transducer \mathcal{M} has a finite set of states Q (including initial states and final states), an input alphabet Σ_1 , a tape alphabet Δ (including \vdash , \dashv , and the elements of Σ_1), an output alphabet Σ_2 , and a finite set of *instructions*. Each instruction is of the form $(p, \sigma, \tau, q, \alpha, \epsilon)$ meaning that if \mathcal{M} is in nonfinal state p and reads tape symbol σ at the current position of the reading head, then it may change σ into the tape symbol τ , change its state into q , write string $\alpha \in \Sigma_2^*$ to the output tape, and move its reading head in direction $\epsilon \in \{-1, 0, +1\}$ (i.e., from position i to position $i + \epsilon$); if $\sigma = \vdash$ ($\sigma = \dashv$), then $\epsilon \neq -1$ ($\epsilon \neq +1$, respectively). The transducer \mathcal{M} realizes the transduction $m \subseteq \Sigma_1^* \times \Sigma_2^*$ such that $(x, y) \in m$ whenever there exists a computation of \mathcal{M} starting with $\vdash x \dashv$ on the input tape, in some initial state with the input head on the leftmost position (reading symbol \vdash), and ending in a final state, while y has been written to the output tape; moreover, we assume w.l.o.g. that all such computations end with the input head on the rightmost position (reading \dashv).

The transduction realized by \mathcal{M} will always be denoted by m . Similarly, the transduction realized by $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}', \dots$ will be denoted m_1, m_2, m', \dots .

As usual, \mathcal{M} is *deterministic* if it has exactly one initial state and does not have two distinct instructions that both start with the same p and σ . Then m is a (partial) function from Σ_1^* to Σ_2^* .

We are only interested in two restrictions of the linear bounded transducer. For the first restriction, we forbid rewriting the symbols of the input tape. To be formal, we require that $\Delta = \Sigma_1 \cup \{\vdash, \dashv\}$ and that $\sigma = \tau$ for each instruction $(p, \sigma, \tau, q, \alpha, \epsilon)$; we will write that instruction as $(p, \sigma, q, \alpha, \epsilon)$. The resulting transducer is the usual *two-way finite-state transducer*, also called two-way nondeterministic generalized sequential machine, abbreviated 2NGSM, or 2DGSM in the deterministic case.

For the second restriction, we forbid an unbounded number of visits to a position of the input tape. Formally, for $k \geq 1$, a computation of \mathcal{M} is *k-visiting* if each of the positions of the input tape is visited at most k times. The transducer \mathcal{M} is *finite-visit* if there is a constant k such that for each pair $(x, y) \in m$ there exists a k -visiting computation on input x with output y . The finite-visit linear bounded transducer is also called a *Hennie machine* (see [13]).

The classes of transductions realized by 2NGSM's and Hennie machines are denoted by 2NGSM and NHM, respectively (and by 2DGSM and DHM in the deterministic case). Obviously, 2DGSM's are always finite-visit (with k the number of states), but that need not be the case for a 2NGSM. The class of transductions realized by finite-visit 2NGSM's (i.e., transducers that are both 2NGSM and Hennie machine) is denoted 2NGSM_{fin}.

The classes of deterministic and nondeterministic MSO definable string transductions are denoted DMSOS and NMSOS, respectively. As observed at the end of the Introduction, we will not need their definition (which can be found in [9]) but we will instead use their characterizations in terms of 2NGSM's and Hennie machines, stated in Proposition 1.1 below (and in Theorem 31 of [9]). The results of [9] and this paper are summarized in the Venn diagram of Fig. 1. Before stating Proposition 1.1 we need some terminology and one additional definition.

For a transduction m and a string x , the image of x under m is $m(x) = \{y \mid (x, y) \in m\}$. For a language L , $m(L) = \bigcup_{x \in L} m(x)$. The domain of transduction m is the language $\{x \mid m(x) \neq \emptyset\}$.

The composition of transductions m_1 and m_2 is $m_1 \circ m_2 = \{(x, z) \mid \exists y : (x, y) \in m_1, (y, z) \in m_2\}$, and for classes of transductions X and Y : $X \circ Y = \{m_1 \circ m_2 \mid m_1 \in X, m_2 \in Y\}$. Moreover, for $n \geq 2$, X^n denotes the n -fold composition of X with itself, and X is closed under composition if $X^2 \subseteq X$. Note the order of application of the transductions in a composition $m_1 \circ m_2$: first m_1 and then m_2 ; thus, for a language L , $(m_1 \circ m_2)(L) = m_2(m_1(L))$.

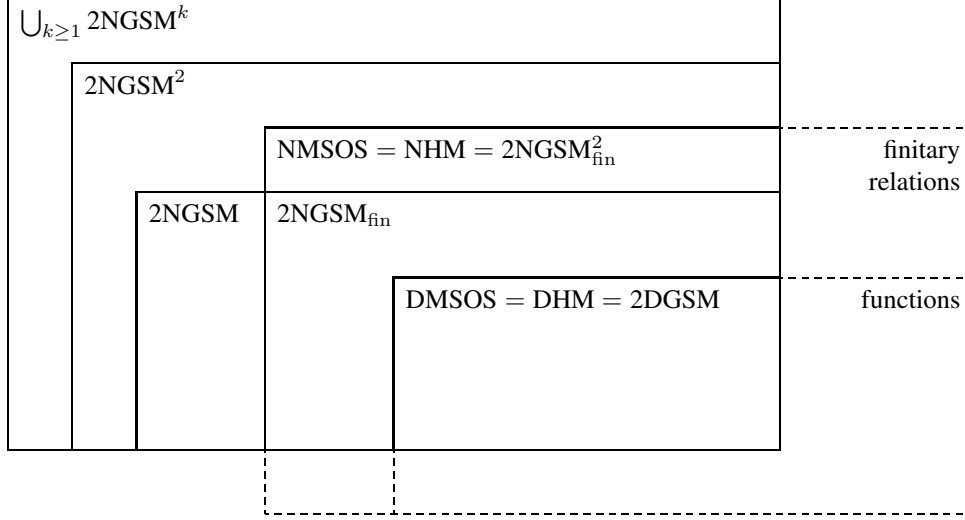


Figure 1. Relationships between our main classes of transductions

Another class of 2NGSM transductions will be needed: the *marked string relabellings* that, for an input string x , apply a nondeterministic string relabelling to the string $\vdash x \dashv$. More precisely, a transduction $m \subseteq \Sigma_1^* \times \Sigma_2^*$ is a marked string relabelling if there exists a binary relation $m_0 \subseteq (\Sigma_1 \cup \{\vdash, \dashv\}) \times \Sigma_2$ such that for every string $x = \sigma_1 \sigma_2 \cdots \sigma_n \in \Sigma_1^*$, $m(x) = m_0(\vdash) \cdot m_0(\sigma_1) \cdot m_0(\sigma_2) \cdots m_0(\sigma_n) \cdot m_0(\dashv)$ where the dot denotes language concatenation. The class of marked string relabellings is denoted MREL .

We now state the main result of [9] (i.e., Theorem 31 of [9]).

Proposition 1.1.

1. $\text{DMSOS} = \text{DHM} = 2\text{DGSM}$.
2. $\text{NMSOS} = \text{NHM} = \text{MREL} \circ 2\text{DGSM} = 2\text{NGSM}_{\text{fin}}^2$.

It is a straightforward property of MSO definable transductions that they are closed under composition, both in the deterministic and nondeterministic case (see, e.g., Proposition 5.5.6 of [5]; see also Propositions 2, 8, and 17 of [9]).

Proposition 1.2. DMSOS and NMSOS are closed under composition.

In this paper, a 2NGSM \mathcal{M} will be called *one-way* if $\epsilon \in \{0, +1\}$ for each instruction $(p, \sigma, q, \alpha, \epsilon)$. Note that the one-way 2NGSM (abbreviated 1NGSM) is more powerful than the usual one-way finite-state transducer, because it has endmarkers around its input and thus can translate the empty string into a nonempty string. As an example, it should be clear that every marked string relabelling can be realized by a 1NGSM . The class of transductions realized by 1NGSM 's is denoted 1NGSM .

The next lemma can be proved by a straightforward product construction.

Lemma 1.1. $2\text{NGSM} \circ 1\text{NGSM} \subseteq 2\text{NGSM}$.

Since two-way finite-state automata recognize the regular languages (as shown in [20]), the domain of a 2NGSM transduction is regular. The same is true for arbitrary compositions of such transductions (as is well known).

Lemma 1.2. For $m \in 2\text{NGSM}^n$, the domain of m is regular.

Proof:

Let \mathcal{M}_1 be a 2NGSM realizing transduction m_1 , and let the domain of transduction m_2 be the regular language R . We will show that the domain of $m_1 \circ m_2$ is regular too. Let \mathcal{M}'_2 be a 1NGSM with $m'_2 = \{(x, x) \mid x \in R\}$. Clearly, the domains of $m_1 \circ m_2$ and $m_1 \circ m'_2$ are the same. By Lemma 1.1 there is a 2NGSM \mathcal{M}'_1 such that $m'_1 = m_1 \circ m'_2$ and its domain is regular by the result of [20]. \square

2. Crossing sequences and visiting sequences

Let \mathcal{M} be a 2DGSM that realizes the transduction $m \subseteq \Sigma_1^* \times \Sigma_2^*$. For strings $u, v \in \Sigma_1^*$ with $m(uv) \neq \emptyset$, the *crossing sequence* of u and v , denoted $CS(u, v)$, is the sequence of states $(p_1, q_1, \dots, p_{n-1}, q_{n-1}, p_n)$ such that, during its computation on input tape $\vdash uv \dashv$, \mathcal{M} first leaves $\vdash u$ entering $v \dashv$ in state p_1 , then from $v \dashv$ enters $\vdash u$ in state q_1 , etc. Thus, it is the sequence of states that \mathcal{M} assumes just after crossing the border between $\vdash u$ and $v \dashv$, alternatingly from left to right and vice versa.

If \mathcal{M} has s states, then the number of crossing sequences of \mathcal{M} is at most $(s+1)^{2s-1}$. That is because in a crossing sequence $(p_1, q_1, \dots, p_{n-1}, q_{n-1}, p_n)$ all p_i are distinct (otherwise \mathcal{M} , being deterministic, would be in a loop), and hence $n \leq s$.

The following basic property of crossing sequences is essentially Theorem 1 of [13]. Think of the computation of \mathcal{M} as a game of ping pong across a given border: then player u_1 does not notice a change of adversary from v_1 to v_2 , provided v_1 and v_2 can react in the same way.

Property (P1):

If $CS(u_1, v_1) = CS(u_2, v_2)$, then $CS(u_1, v_2) = CS(u_1, v_1)$, and, more generally for $u_1 = u'_1 u''_1$, $CS(u'_1, u''_1 v_2) = CS(u'_1, u''_1 v_1)$.

Symmetrically, if $CS(u_1, v_1) = CS(u_2, v_2)$, then $CS(u_2, v_1) = CS(u_1, v_1)$, and, more generally for $v_1 = v'_1 v''_1$, $CS(u_2 v'_1, v''_1) = CS(u_1 v'_1, v''_1)$.

For strings $u, v, w \in \Sigma_1^*$ with $m(uvw) \neq \emptyset$ and $v \neq \lambda$ (the empty string), the *visiting sequence* of u, v, w , denoted $VS(u, v, w)$, is the sequence of 5-tuples (d, p, e, q, α) , with $d, e \in \{-1, +1\}$, $p, q \in Q$, and $\alpha \in \Sigma_2^*$, such that \mathcal{M} , during its computation on input tape $\vdash uvw \dashv$, enters the segment v in direction d into state p , and after some moves ‘inside’ v leaves the segment in direction e into state q , producing output α during the moves inside v (including the exit move), for each of the 5-tuples, in the given order. Thus, each 5-tuple represents a ‘visit’ to the segment v , and the sequence of these 5-tuples represents the sequence of consecutive visits to v . An example will be given in Example 2.1.

A visiting sequence is *bare* if $\alpha = \lambda$ for all its 5-tuples (d, p, e, q, α) . Clearly, the number of visiting sequences of \mathcal{M} can be infinite. However, if \mathcal{M} has s states, then the number of *bare* visiting sequences is at most $(2s+1)^{4s}$ (because, again, \mathcal{M} cannot be in a loop).

Now think of a game of ping pong with three players, two tables and one ball; one player plays against the other two, on both tables. These two players, u_1 and w_1 will not notice a change of their

common adversary v_1 to an equally versatile adversary v_2 . The noises of the strokes, heard by the public, will also be the same.

Property (P2):

If $VS(u_1, v_1, w_1) = VS(u_2, v_2, w_2)$, then $m(u_1v_1w_1) = m(u_2v_2w_2)$.

Note that $VS(u, v, w)$ uniquely determines $CS(u, vw)$ and $CS(uv, w)$. This is not true vice versa (as shown in Example 2.1), unless the segment v is fixed (as stated in the next property).

Property (P3):

If $CS(u_1, vw_1) = CS(u_2, vw_2)$ and $CS(u_1v, w_1) = CS(u_2v, w_2)$, then $VS(u_1, v, w_1) = VS(u_2, v, w_2)$.

Thus, player v does not notice a change of adversaries, when each new adversary plays against v in the same way as the old one.

Example 2.1. The 2DGSM \mathcal{M} has states 1 to 6, initial state 1, final state 6, and instructions $(p, \sigma, q, \alpha, \epsilon)$ where the triple q, α, ϵ for each pair $p \in \{1, 2, \dots, 5\}$, $\sigma \in \{\vdash, a, b, \dashv\}$ is given in the following matrix.

	1	2	3	4	5
\vdash	1, λ , +1	3, b , 0	3, λ , +1	5, b , 0	5, λ , +1
a	1, a , +1	2, a , -1	3, a , +1	4, a , -1	5, a , +1
b	2, λ , -1	4, λ , -1	1, λ , +1	5, λ , +1	3, λ , +1
\dashv	2, c , 0	2, λ , -1	4, c , 0	4, λ , -1	6, λ , 0

On each segment of a 's of the input \mathcal{M} makes five passes in states 1 to 5, each in alternate directions, while copying the symbols to the output.

On a symbol b the machine does not generate output, but it performs a permutation of the order in which the two neighbouring segments of a 's are read. This is best explained by looking at the computations on the input strings $a^3b^i a^2$, $i = 0, 1, 2, 3$ as depicted in Figure 2. The output strings for these inputs are given in the following table.

input string	output string
$a^5 = a^3b^0 a^2$	$a^5ca^5ba^5ca^5ba^5 = a^3(a^2ca^2)(a^3ba^3)(a^2ca^2)(a^3ba^3)a^2$
$a^3b^1 a^2$	$a^6ba^5ca^5ba^5ca^4 = a^3(a^3ba^3)(a^2ca^2)(a^3ba^3)(a^2ca^2)a^2$
$a^3b^i a^2, i \geq 2$	$a^6ba^6ba^5ca^4ca^4 = a^3(a^3ba^3)(a^3ba^3)(a^2ca^2)(a^2ca^2)a^2$

It can easily be seen from Figure 2 that the crossing sequences are all the same: $CS(a^3, b^i a^2) = CS(a^3b^i, a^2) = (1, 2, 3, 4, 5)$, for $i = 0, 1, 2, 3$. The visiting sequences are the following (with +, - instead of +1, -1 for the directions):

$$\begin{aligned}
 VS(a^3, b, a^2) &= (+, \mathbf{1}, -, \mathbf{2}, \lambda) \\
 &\quad (+, \mathbf{3}, +, \mathbf{1}, \lambda) \\
 &\quad (-, \mathbf{2}, -, \mathbf{4}, \lambda) \\
 &\quad (+, \mathbf{5}, +, \mathbf{3}, \lambda) \\
 &\quad (-, \mathbf{4}, +, \mathbf{5}, \lambda)
 \end{aligned}$$

and

$$\begin{aligned}
 VS(a^3, b^2, a^2) = VS(a^3, b^3, a^2) = & (+, \mathbf{1}, -, \mathbf{2}, \lambda) \\
 & (+, \mathbf{3}, -, \mathbf{4}, \lambda) \\
 & (+, \mathbf{5}, +, 1, \lambda) \\
 & (-, 2, +, 3, \lambda) \\
 & (-, 4, +, 5, \lambda),
 \end{aligned}$$

where the states that contribute to $CS(a^3, b^i a^2)$ are in boldface, the remaining ones contributing to $CS(a^3 b^i, a^2)$. Note that all these visiting sequences are bare.

The next lemma is a ‘pumping lemma’, cf. Corollary 1 of [13]. An example is given after the proof.

Lemma 2.1. If $CS(u, vw) = CS(uv, w)$ and $VS(u, v, w)$ is not bare, then $m(uv^*w)$ is infinite.

Proof:

First we prove, by repeated use of property (P1), that the crossing sequences $CS(uv^i, v^{n-i}w)$ are all the same for $n \geq 1$ and $0 \leq i \leq n$, i.e., they are all equal to $CS(u, vw) = CS(uv, w)$. This is given for $n = 1$. Assuming by induction that it holds for n , we now prove it for $n + 1$.

Consider first the case that $i = 0$. By the induction hypothesis, $CS(uv, w) = CS(u, v^n w)$. Changing ‘adversary’ w of uv into $v^n w$ we obtain that $CS(u, vv^n w) = CS(u, vw)$ by property (P1) (with $u'_1 = u$, $u''_1 = v$, $u_2 = u$, $v_1 = w$, and $v_2 = v^n w$).

The case that $i = n + 1$ is symmetric. Changing adversary u of vw into uv^n , we obtain from $CS(u, vw) = CS(uv^n, w)$ that $CS(uv^n v, w) = CS(uv, w)$ by the symmetric version of property (P1) (with $u_1 = u$, $v'_1 = v$, $v''_1 = w$, $u_2 = uv^n$, and $v_2 = w$).

For $i = 1, \dots, n$ we get, by induction, $CS(uv^i, v^{n-i}w) = CS(uv^{i-1}, v^{n-(i-1)}w)$. Hence, by property (P1), $CS(uv^i, v^{n-(i-1)}w) = CS(uv^i, v^{n-i}w)$, and note that $n - (i - 1) = (n + 1) - i$.

This shows the result above. Property (P3) now implies that $VS(uv^i, v, v^j w) = VS(u, v, w)$ for all $i, j \geq 0$. Since $VS(u, v, w)$ is not bare, this implies that for input string $uv^n w$ at least one output symbol is produced by the transducer \mathcal{M} on each segment v , and hence the length of $m(uv^n w)$ is at least n . This shows that $m(uv^*w)$ is infinite. \square

As an example, change the instruction $(1, b, 2, \lambda, -1)$ of the 2DGSM \mathcal{M} of Example 2.1 into $(1, b, 2, d, -1)$. Then $VS(a^3, b, a^2)$ is not bare any more, as its first 5-tuple now equals $(+, 1, -, 2, d)$. The output strings are changed as follows:

input string	output string
$a^3 b^1 a^2$	$a^3 d(a^3 b a^3)(a^2 c a^2)(a^3 b a^3)(a^2 c a^2) a^2$
$a^3 b^i a^2, i \geq 2$	$a^3 d(a^3 b a^3) d(a^3 b a^3) d^{i-2}(a^2 c a^2)(a^2 c a^2) a^2$

and hence $m(a^3 b^* a^2)$ is indeed infinite.

3. Finitary compositions

A transduction m is *finitary on a language L* if $m(x)$ is finite for every $x \in L$.

In this section we develop two characterizations for the composition $m_1 \circ m_2$ to be finitary on L , where \mathcal{M}_1 is a 2NGSM and \mathcal{M}_2 is a 2DGSM (and we assume that m_2 is defined for every output of m_1).

The first one is in terms of a finite-visit property, defined as follows. A 2NGSM \mathcal{M} is *finite-visit on a language L with respect to a function f* (that is assumed to be total on the range of m) if there is a constant k such that for every $(x, y) \in m$ with $x \in L$ there is a k -visiting computation of \mathcal{M} on input x that has output y' with $f(y') = f(y)$. We will show that $m_1 \circ m_2$ is finitary on L if and only if \mathcal{M}_1 is finite-visit on L with respect to m_2 .

The second characterization says that $m_1 \circ m_2$ is finitary on L if and only if \mathcal{M}_1 is not repetitive on L with respect to \mathcal{M}_2 , where the, rather specific, concept of repetitiveness is defined as follows. We say that a 2NGSM \mathcal{M}_1 is *repetitive* on a language L with respect to a 2DGSM \mathcal{M}_2 if there is a string $x \in L$ such that \mathcal{M}_1 has a computation on input x that visits the same position of $\vdash x \dashv$ in the same state twice, with the following property: if u is the output of the computation before the first visit, v is the output between the two visits, and w is the output after the second visit, then $m_2(uvw) \neq \emptyset$, $v \neq \lambda$, $CS(u, vw) = CS(uv, w)$, and $VS(u, v, w)$ is not bare. Of course, CS and VS are meant to refer to \mathcal{M}_2 .

Lemma 3.1. Let \mathcal{M}_1 be a 2NGSM and \mathcal{M}_2 a 2DGSM, such that m_2 is defined for every output of m_1 . The following three statements are equivalent, for every language L .

1. $m_1 \circ m_2$ is finitary on L .
2. \mathcal{M}_1 is not repetitive on L with respect to \mathcal{M}_2 .
3. \mathcal{M}_1 is finite-visit on L with respect to m_2 .

Proof:

(1) \Rightarrow (2). Suppose that \mathcal{M}_1 has such a computation, on input $x \in L$. Clearly, by looping, \mathcal{M}_1 can produce the output uv^nw on input x , for every n . Hence, by Lemma 2.1, $m_2(m_1(x))$ is infinite. This contradicts the fact that $m_1 \circ m_2$ is finitary on L .

(3) \Rightarrow (1). The length of the output of a k -visiting computation is linear in the length of the input. Thus, for every x , the set of all y' such that \mathcal{M}_1 has a k -visiting computation on input x with output y' , is finite. Consequently, for $x \in L$, the image of this set under m_2 is finite too.

(2) \Rightarrow (3). This is the central argument of the paper. Let \mathcal{M}_2 have s states, let γ be an upper bound on the number of crossing sequences of \mathcal{M}_2 , and let β be an upper bound on the number of bare visiting sequences of \mathcal{M}_2 (see Section 2). We claim that \mathcal{M}_1 is k -visit on L with respect to m_2 , where $k = s \cdot \gamma \cdot (\beta + 1)$.

Suppose that, for some input $x \in L$ and some position of $\vdash x \dashv$, \mathcal{M}_1 has a computation on x that visits that position more than k times (with output y). Then there is a state of \mathcal{M}_1 such that it visits that position in that state more than $\gamma \cdot (\beta + 1)$ times. For those visits, consider the crossing sequences $CS(y_i, y'_i)$ of \mathcal{M}_2 , where y_i (y'_i) is the output produced by the computation before (after) the i -th visit. Since there are more than $\gamma \cdot (\beta + 1)$ such visits, there must be $\beta + 2$ indices i such that the $CS(y_i, y'_i)$ are all the same. Consider those $\beta + 2$ visits in particular, and let $t = \beta + 1$. Then the output produced by the computation can be written $y = uv_1v_2 \cdots v_tv$, where u is produced before the first visit, v_i

during the excursion between visit i and visit $i + 1$, and w after the last visit. The crossing sequences $CS(uv_1 \cdots v_{i-1}, v_i \cdots v_t w)$ are all the same, for $1 \leq i \leq t + 1$. Our aim is now to show that there exists another computation of \mathcal{M}_1 on x that visits the given position less times and produces an output y' with $m_2(y') = m_2(y)$. That shows that there exists an appropriate computation that visits the position at most k times, and hence, by repeating the procedure for every position of $\vdash x \dashv$, that there exists an appropriate k -visiting computation. (Note that, for this to work, we should be careful that the number of visits to the other positions does not increase; this will hold because the new computation is obtained from the old one by skipping the excursions between two or more consecutive visits to the given position in the given state.)

If $v_i = \lambda$ for some i , then another computation with the same output y is obtained by skipping the excursion between visit i and visit $i + 1$, and we are ready. Now assume that $v_i \neq \lambda$ for all i . Since \mathcal{M}_1 is not repetitive on L with respect to \mathcal{M}_2 , $VS(u, v_1 \cdots v_{i-1}, v_i \cdots v_t w)$ is bare, for every $2 \leq i \leq t + 1$; note that $v_1 \cdots v_{i-1} \neq \lambda$. Thus, since $t > \beta$ (and β is an upper bound on the number of bare visiting sequences of \mathcal{M}_2), there exist i and j with $2 \leq i < j \leq t + 1$ such that

$$VS(u, v_1 \cdots v_{i-1}, v_i \cdots v_t w) = VS(u, v_1 \cdots v_{j-1}, v_j \cdots v_t w).$$

Consider now the computation of \mathcal{M}_1 that is obtained by skipping the excursions between visits i and j . It produces output $y' = uv_1 \cdots v_{i-1}v_j \cdots v_t w$ and, by property (P2), $m_2(y') = m_2(y)$. \square

4. Collapse

In this section we show that a composition of 2NGSM transductions is (nondeterministic) MSO definable if and only if it is finitary. In one direction this is obvious because all transductions in NMSOS are finitary (e.g., by Proposition 1.1). For the other direction we need one more lemma.

Lemma 4.1. If a 2NGSM \mathcal{M} is finite-visit on L with respect to f , then there exists $m' \in \text{NMSOS}$ such that $f(m'(x)) = f(m(x))$ for all $x \in L$.

Proof:

Using Proposition 1.1, we show the existence of a nondeterministic Hennie machine \mathcal{M}' such that m' satisfies the requirement. To this aim it suffices to construct \mathcal{M}' in such a way that it simulates the k -visiting computations of \mathcal{M} . This is easy: \mathcal{M}' simulates \mathcal{M} and simultaneously counts the visits to each square, on a separate track, up to k . To be more precise, every instruction $(p, \sigma, q, \alpha, \epsilon)$ of \mathcal{M} is simulated by the instructions $(p, \langle \sigma, i \rangle, \langle \sigma, i+1 \rangle, q, \alpha, \epsilon)$ of \mathcal{M}' , for $0 \leq i < k$ (where $\langle \sigma, 0 \rangle$ is identified with σ). \square

The next theorem is the first result of this paper. For $n = 1$ it was shown in Lemma 28 of [9].

Theorem 4.1. If $m \in 2\text{NGSM}^n$ is finitary, then $m \in \text{NMSOS}$.

Proof:

We show by induction on n a slightly stronger statement: if $m \in 2\text{NGSM}^n \circ \text{NMSOS}$ is finitary, then $m \in \text{NMSOS}$. It is trivial for $n = 0$. Now consider $m = m_0 \circ m_1 \circ m_2$ with $m_0 \in 2\text{NGSM}^n$, $m_1 \in 2\text{NGSM}$, and $m_2 \in \text{NMSOS}$. Since $\text{NMSOS} = \text{MREL} \circ 2\text{DGSM}$ by Proposition 1.1 and

$2\text{NGSM} \circ \text{MREL} \subseteq 2\text{NGSM}$ by Lemma 1.1 (because every marked string relabelling can be realized by a 1NGSM), we may assume that $m_2 \in 2\text{DGSM}$.

Let \mathcal{M}_1 be a 2NGSM that realizes m_1 . Since the domain of m_2 is a regular language by Lemma 1.2, we can modify \mathcal{M}_1 in such a way that it checks that its output is in that regular language (see the argument in the proof of Lemma 1.2, where \mathcal{M}_1 is modified into \mathcal{M}'_1). Hence we may additionally assume that m_2 is defined for every output of m_1 .

Since m is finitary, $m_1 \circ m_2$ is finitary on the range of m_0 . Hence, by Lemma 3.1, \mathcal{M}_1 is finite-visit on the range of m_0 with respect to m_2 . And so, by Lemma 4.1, there exists $m'_1 \in \text{NMSOS}$ such that $m_2(m'_1(x)) = m_2(m_1(x))$ for every x in the range of m_0 . Hence $m = m_0 \circ m'_1 \circ m_2$. Consequently, since $\text{NMSOS} \circ 2\text{DGSM} \subseteq \text{NMSOS}$ by Propositions 1.1 and 1.2, $m \in 2\text{NGSM}^n \circ \text{NMSOS}$. \square

A string transduction m is of *linear size increase* if the length of the output is linear in the length of the input, i.e., if there is a constant c such that $|y| \leq c \cdot |x|$ for every $(x, y) \in m$. We now conclude that a composition of 2NGSM transductions is finitary if and only if it is of linear size increase. In the if direction this is immediate. In the only-if direction it follows from Theorem 4.1 because all transductions in NMSOS are of linear size increase: this is obvious from the definition of NMSOS , cf. [9], or from Proposition 1.1 and the definition of $2\text{NGSM}_{\text{fin}}$. Hence, a composition of 2NGSM transductions is MSO definable if and only if it is of linear size increase (which is a result similar to the one in [11, 16] for tree transductions).

5. Decidability

In this section we prove that it is decidable for a sequence of 2NGSM transductions whether or not their composition is (nondeterministic) MSO definable. Let us observe here that all the previous propositions, lemmas, and theorems are effective.

For a sequence of 2NGSM 's $\mathcal{M} = (\mathcal{M}_1, \dots, \mathcal{M}_n)$, we denote the composition $m_1 \circ \dots \circ m_n$ by m .

Lemma 5.1. It is decidable for a sequence \mathcal{M}_0 of 2NGSM 's, a 2NGSM \mathcal{M}_1 , and a 2DGSM \mathcal{M}_2 , whether or not \mathcal{M}_1 is repetitive on the range of m_0 with respect to \mathcal{M}_2 .

Proof:

Let $\overline{\mathcal{M}}$ be the (one-way) 2NGSM that, on input x , outputs the string $\vdash x \dashv$ in which exactly one symbol is barred (nondeterministically). More precisely, for $x \in \Sigma_0^*$ (where Σ_0 is the output alphabet of m_0), $\overline{m}(x) = \{x_1 \bar{\sigma} x_2 \mid x_1 \sigma x_2 = \vdash x \dashv \text{ with } x_1 \in (\Sigma_0 \cup \{\vdash\})^*, \sigma \in \Sigma_0 \cup \{\vdash, \dashv\}, \text{ and } x_2 \in (\Sigma_0 \cup \{\dashv\})^*\}$.

Let \mathcal{M}'_1 be the 2NGSM that, on input of such a barred version of $\vdash x \dashv$, simulates \mathcal{M}_1 on input x and nondeterministically outputs $u\bar{v}w$ instead of uvw as described in the definition of 'repetitive' (without the conditions on CS and VS). To be precise, \mathcal{M}'_1 outputs $u\bar{v}w$ (where \bar{v} is obtained from v by barring each of its symbols) if and only if \mathcal{M}_1 has a computation on input tape $\vdash x \dashv$ that visits the barred position of $\vdash x \dashv$ in the same state twice, where u is the output of the computation before the first visit, v is the output between the two visits, and w is the output after the second visit.

Let \mathcal{M}'_2 be the 2DGSM that, on input $u\bar{v}w$, simulates \mathcal{M}_2 on input uvw and checks that $v \neq \lambda$, $CS(u, v\bar{v}w) = CS(uv, w)$, and $VS(u, v, w)$ is not bare. Note that \mathcal{M}'_2 can compute these crossing sequences and visiting sequence during its simulation of \mathcal{M}_2 , storing them in its state.

It should be clear that \mathcal{M}_1 is repetitive on the range of m_0 with respect to \mathcal{M}_2 if and only if the domain of $m_0 \circ \bar{m} \circ m'_1 \circ m'_2$ is nonempty. By Lemma 1.2 this domain is regular and its emptiness can be checked. \square

The next theorem is the second result of this paper.

Theorem 5.1. It is decidable for a sequence \mathcal{M} of 2NGSM's whether or not $m \in \text{NMSOS}$.

Proof:

As observed before, all NMSOS transductions are finitary (e.g., by Proposition 1.1). Hence, by Theorem 4.1, $m \in \text{NMSOS}$ if and only if m is finitary. Thus, we may instead prove the decidability of finitariness.

As in the proof of Theorem 4.1, we show by induction on n a slightly stronger statement: it is decidable for a sequence \mathcal{M}_0 of n 2NGSM's and a device \mathcal{M}_2 with $m_2 \in \text{NMSOS}$ whether or not $m_0 \circ m_2$ is finitary. This is obvious for $n = 0$ because all transductions in NMSOS are finitary. Now consider a 2NGSM \mathcal{M}_1 . We wish to test whether or not $m = m_0 \circ m_1 \circ m_2$ is finitary. As in the proof of Theorem 4.1 we may assume that \mathcal{M}_2 is a 2DGSM and that m_2 is defined for every output of m_1 .

Using Lemma 5.1, we first test whether or not \mathcal{M}_1 is repetitive on the range of m_0 with respect to \mathcal{M}_2 . If so, then, by Lemma 3.1, $m_1 \circ m_2$ is not finitary on the range of m_0 , and hence m is not finitary. Now assume that it is *not* repetitive. Then, again by Lemma 3.1, \mathcal{M}_1 is finite-visit on the range of m_0 with respect to m_2 . And so, as in the proof of Theorem 4.1, $m \in 2\text{NGSM}^n \circ \text{NMSOS}$. \square

Thus, it is decidable for a sequence of 2NGSM's $\mathcal{M}_1, \dots, \mathcal{M}_n$ whether or not $m_1 \circ \dots \circ m_n \in \text{NMSOS}$. For $n = 1$ this was shown in Theorem 30 of [9].

Acknowledgement. We are grateful to Antoni Mazurkiewicz for having crossed our paths so many times, in particular for his visiting sequence to Leiden. As a result, his work has been an inspiration to us. One of the first manuscripts of one of us ([6]) was based on Antoni's early work on the theory of correctness ([17]). Another theory, the theory of traces, was initiated by Antoni ([18, 19]), and we enjoyed working on the intricacies of traces ([1, 2, 15, 14]).

References

- [1] Aalbersberg, IJ. J., Engelfriet, J.: *The recognition of trace languages*, Tech. Report 86-18, Leiden University, 1986.
- [2] Aalbersberg, IJ. J., Hoogeboom, H. J.: Characterizations of the decidability of some problems for regular trace languages, *Mathematical Systems Theory*, **22**, 1989, 1–19.
- [3] Bloem, R., Engelfriet, J.: A comparison of tree transductions defined by monadic second order logic and by attribute grammars, *Journal of Computer and System Sciences*, **61**, 2000, 1–50.
- [4] Courcelle, B.: Monadic second-order definable graph transductions: a survey, *Theoretical Computer Science*, **126**, 1994, 53–75.
- [5] Courcelle, B.: The expression of graph properties and graph transformations in monadic second-order logic, in: *Handbook of Graph Grammars and Computing by Graph Transformation, vol.1: Foundations* (G. Rozenberg, Ed.), World Scientific, 1997, 313–400.

- [6] Engelfriet, J.: *Proving correctness by head and tail relations*, Manuscript, Twente University of Technology, 1972.
- [7] Engelfriet, J.: Three hierarchies of transducers, *Mathematical Systems Theory*, **15**, 1982, 95–125.
- [8] Engelfriet, J.: Iterated stack automata and complexity classes, *Information and Computation*, **95**, 1991, 21–75.
- [9] Engelfriet, J., Hoogeboom, H. J.: MSO definable string transductions and two-way finite-state transducers, *ACM Transactions on Computational Logic*, **2**, 2001, 216–254.
- [10] Engelfriet, J., Maneth, S.: Macro tree transducers, attribute grammars, and MSO definable tree translations, *Information and Computation*, **154**, 1999, 34–91.
- [11] Engelfriet, J., Maneth, S.: Macro tree translations of linear size increase are MSO definable, *SIAM Journal on Computing*, **32**, 2003, 950–1006.
- [12] Greibach, S. A.: Hierarchy theorems for two-way finite state transducers, *Acta Informatica*, **11**, 1978, 89–101.
- [13] Hennie, F. C.: One-tape, off-line Turing machine computations, *Information and Control*, **8**, 1965, 553–578.
- [14] Hoogeboom, H. J., Muscholl, A.: The code problem for traces - improving the boundaries, *Theoretical Computer Science*, **172**, 1997, 309–321.
- [15] Hoogeboom, H. J., Rozenberg, G.: Dependence graphs, in: *The Book of Traces* (V. Diekert, G. Rozenberg, Eds.), World Scientific, 1995, 43–68.
- [16] Maneth, S.: The macro tree transducer hierarchy collapses for functions of linear size increase, in: *Proc. FSTTCS'2003* (P. K. Pandya, J. Radhakrishnan, Eds.), Lecture Notes in Computer Science 2914, Springer-Verlag, 2003, 326–337.
- [17] Mazurkiewicz, A. W.: Proving algorithms by tail functions, *Information and Control*, **18**, 1971, 220–226.
- [18] Mazurkiewicz, A. W.: *Concurrent program schemes and their interpretations*, DAIMI PB-78, Aarhus University, 1977.
- [19] Mazurkiewicz, A. W.: Trace theory, in: *Petri Nets: Central Models and Their Properties* (W. Brauer, W. Reisig, G. Rozenberg, Eds.), Lecture Notes in Computer Science 255, Springer-Verlag, 1987, 279–324.
- [20] Shepherdson, J. C.: The reduction of two-way automata to one-way automata, *IBM Journal of Research and Development*, **3**, 1959, 198–200.