MSO Definable String Transductions and Two-Way Finite-State Transducers

JOOST ENGELFRIET and HENDRIK JAN HOOGEBOOM
Universiteit Leiden

We extend a classic result of Büchi, Elgot, and Trakhtenbrot: MSO definable string transductions, i.e., string-to-string functions that are definable by an interpretation using monadic second-order (MSO) logic, are exactly those realized by deterministic two-way finite-state transducers, i.e., finite-state automata with a two-way input tape and a one-way output tape. Consequently, the equivalence of two MSO definable string transductions is decidable. In the nondeterministic case however, MSO definable string transductions, i.e., binary relations on strings that are MSO definable by an interpretation with parameters, are incomparable to those realized by nondeterministic two-way finite-state transducers. This is a motivation to look for another machine model, and we show that both classes of MSO definable string transductions are characterized in terms of Hennie machines, i.e., two-way finite-state transducers that are allowed to rewrite their input tape, but may visit each position of their input only a bounded number of times.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; *Relations between models*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages

General Terms: Theory

Additional Key Words and Phrases: Büchi, Elgot, Hennie machine, interpretation, monadic second-order logic, string transductions, Trakhtenbrot, two-way finite-state transducers

1. INTRODUCTION

It is always a pleasant surprise when two formalisms, introduced with different motivations, turn out to be equally powerful, as this indicates that the underlying concept is a natural one. Additionally, this means that notions and tools from one formalism can be made use of within the other, leading to a better understanding of the formalisms under consideration. A famous example is of course the regular languages (regular sets of strings [Yu 1997]) that can be defined using a computational formalism (finite-state automata, either deterministic or nondeterministic), but also have well-known grammatical (right-linear grammars), operational (rational operations), algebraic (congruences of finite index),

An extended abstract of this paper was presented at ICALP'99 [Engelfriet and Hoogeboom 1999].
Authors' address: Universiteit Leiden, Leiden Institute of Advanced Computer Science, P.O. Box 9512, 2300 RA Leiden, The Netherlands, `http://www.liacs.nl/CS/TCS/`

and in particular logical (monadic second-order logic of one successor) characterizations [McCulloch and Pitts 1943; Rabin and Scott 1959; Chomsky 1956; Kleene 1956; Myhill 1957; Nerode 1958; Büchi 1960; Elgot 1961; Trakhtenbrot 1962].

In this paper we study "regular" string transductions (i.e., functions from strings to strings or, more generally, binary relations on strings), rather than regular languages, and we obtain the equivalence of particular computational and logical formalisms, modestly following in the footsteps of Büchi [1960], Elgot [1961], and Trakhtenbrot [1962]. Their original work demonstrates how a logical formula may effectively be transformed into a finite-state automaton accepting the language specified by the formula when interpreted over finite sequences. From a logical point of view this makes it possible to use finite state automata as a tool to solve decidability questions concerning the validity of formulas. From a computer science point of view it shows how to relate the *specification* of a system behavior (as given by the formula) to a possible *implementation* (as the finite-state behavior of an automaton). In recent years much effort has been put into transforming these initial theoretical results into software tools for the verification of finite-state systems, *model checking*; see the monographs Kurshan [1994], and Clarke et al. [1999].

Generalizations of the result of Büchi, Elgot, and Trakhtenbrot include infinite strings [Büchi 1962], trees [Doner 1970; Thatcher and Wright 1968], infinite trees [Rabin 1969], graphs of bounded tree-width [Lapoire 1998], traces (a syntactic model for concurrency) [Ebinger 1995], texts (strings with an additional ordering) [Hoogeboom and ten Pas 1997], and tree-to-tree transductions [Bloem and Engelfriet 2000; Engelfriet and Maneth 1999]. We refer to Thomas [1997b] for an overview of the study of formal languages within the framework of mathematical logic.

We give a short description of the two formalisms of "regular" string transductions that we study in this paper. We mainly consider the deterministic case, i.e., functions rather than relations.

The monadic second-order (MSO) logic of one successor is a logical framework that allows one to specify string properties using quantification over sets of positions in the string. As stated above, Büchi, Elgot, and Trakhtenbrot proved that the languages specified by MSO definable properties are exactly the regular languages. The logic has a natural generalization to graphs, with quantification over sets of nodes, and predicates referring to node labels and edge labels. It is used to define graph-to-graph transductions, by specifying the edges of the output graph in terms of MSO properties of (copies of) a given input graph. In other words, we consider graph transductions that are defined by an appropriate MSO version of the notion of interpretation of logical structures, well known in model theory see, e.g., Hodges [1993, Chapter 5], Rabin [1977], and Seese [1992, Section 6]. (Observe that the graph-to-graph transduction we consider here interprets the output graph in the disjoint sum of a fixed number of copies of the input graph, rather than in a product of copies of the input graph; see Section 4.)

These MSO definable graph transductions play an important role in the theory of graph grammars (for a survey, see Courcelle [1994;1997], and Engelfriet

[1997]). In fact, the two main classes of context-free graph languages can be obtained by applying MSO definable graph transductions to regular tree languages [Engelfriet and van Oostrom 1997; Courcelle and Engelfriet 1995] (see also Courcelle [1992], where the result of Engelfriet and van Oostrom [1997] is generalized to arbitrary logical structures).

Here we consider MSO definable string transductions, i.e., the restriction of MSO definable graph transductions to linear input and output graphs. As an example, it is easy to define the string transduction $\{(w, ww) \mid w \in \{a, b\}^*\}$ by taking two copies of the linear graph representing w and specifying that the last symbol of the first copy is connected to the first symbol of the second copy. As usual for interpretations [Hodges 1993], the class of MSO definable (string) transductions is closed under composition, and the class of regular languages is closed under inverse MSO definable transductions (recall that regular is equivalent to MSO definable) see, e.g., Courcelle [1994]. From this last general result one may infer a large number of specific closure properties of the regular languages, such as closure under the "root" operation $\sqrt{K} = \{w \mid ww \in K\}$. It should be clear from the above example that the regular languages are not closed under MSO definable transductions.

A two-way finite-state *transducer* (or two-way generalized sequential machine, 2GSM) is a finite-state automaton equipped with a two-way input tape, and a one-way output tape. Such a transducer may freely move over its input tape, and may typically reverse or copy parts of its input string. It is, e.g., straightforward to construct a transducer realizing the relation $\{(w, ww) \mid w \in \{a, b\}^*\}$: the machine traverses its tape from left to right, copying each of the symbols to the output, and when it reaches the left end of the input it returns to the beginning to make another copy. Thus, the regular languages are not closed under 2GSM transductions, as opposed to their closure under one-way GSM transductions.

However, it is well known [Rabin and Scott 1959; Shepherdson 1959; Hopcroft and Ullman 1979] that two-way finite-state automata accept only regular languages, and consequently (using a elementary direct product construction) the regular languages are closed under inverse 2GSM transductions. It is maybe less well known that the (deterministic) 2GSM transductions are closed under composition [Chytil and Jákl 1977]. This result is used as a powerful tool in this paper.

Apart from these similar closure properties there is more evidence in the literature that indicates the close connection between MSO definable transductions and 2GSM transductions. First, various specific 2GSM transductions were shown to be MSO definable, such as one-way GSM transductions, mirror image, and mapping the string w onto w^n (for fixed n), cf. Courcelle [1997, Proposition 5.5.3]. Second, returning to the theory of graph grammars, it is explained in Engelfriet [1997, pp. 192–198] that the ranges of MSO definable string transductions are equal to the (string) languages defined by linear context-free graph grammars, which, by a result of Engelfriet and Heyker [1991], equal the ranges of 2GSM transductions. Consequently, the two classes of transductions we consider have the same generative power (on regular input). This, however, does not answer the question whether they are the same class of transductions (cf. Section 6 of

Courcelle [1994]). In this paper we answer this question positively (in the deterministic case). Thus, string transductions that are *specified* as an MSO logical interpretation can be *implemented* on 2GSM's, and vice versa.

Our paper is organized as follows.

In a preliminary section we mainly recall notions and notations regarding graphs, in particular MSO logic for graphs and strings. Moreover, we recall the usual, natural representation of (nonempty) strings as linear graphs that allows a transparent interpretation of strings and languages within the setting of the MSO logic for graphs.

In Section 3 we study *two-way machines*, in particular two-way generalized sequential machines. We extend the basic model by allowing the machines to "jump" to new positions on the tape (not necessarily adjacent to the present position) as specified by an MSO formula that is part of the instructions. This "hybrid" model (in between logic and machine) facilitates the proof of our main result. We consider yet another variant of the 2GSM which allows "regular look-around," i.e., the ability to test the strings to the left and to the right of the reading head for membership in a regular language. The equivalence of the basic 2GSM model and our two extended models (in the deterministic case) is demonstrated using the closure of 2GSM transductions under composition and using Büchi, Elgot, and Trakhtenbrot's result for regular languages.

In Section 4 we recall the definition of (deterministic) MSO definable graph transduction, and restrict that general notion to MSO definable string transductions by considering graph representations for strings. Instead of the representation of Section 2, we use an alternative, natural and well-known, graph representation for strings. Again it uses linear graphs, but now with labels on the edges rather than on the nodes to represent the symbols of the string. This representation allows us to treat the empty string in a proper way.

The main result of the paper is presented as Theorem 13: the equivalence of the (deterministic) MSO definable string transductions from Section 4, and the (deterministic) 2GSM's from Section 3. Section 5 contains the proof of this result. In order to transform a 2GSM into the MSO formalism we consider the "computation space" of a 2GSM on a given input. This is the graph which has a node for each configuration, consisting of an input tape position and a state, of the 2GSM. These nodes are connected by edges representing the possible moves of the 2GSM. The transduction is then decomposed into two transductions, each of which is shown to be MSO definable. First the computation space is defined in terms of k copies of the input string, where k is the number of states of the 2GSM; then the computation path for the input (and its resulting output string) is recovered from the computation graph. One implication of the main result then follows by the closure of MSO definable (graph!) transductions under composition. The reverse implication is obtained by transforming an MSO definable string transduction into a 2GSM equipped with MSO instructions, the tool we introduced in Section 3. As a corollary of our main result we obtain the decidability of the equivalence problem for deterministic MSO definable string transductions. In the nondeterministic case this problem is undecidable because it is undecidable for one-way GSM's [Griffiths 1968], which are (effectively) MSO definable, as observed above.

In Section 6 we study nondeterminism. This feature is realized in MSO definable transductions by the use of parameters: free set variables in the MSO formulas of the interpretation [Hodges 1993; Rabin 1977; Courcelle 1997]. The output of the transduction for a given input may then vary for different valuations of these parameters. These nondeterministic transductions are closed under composition, as opposed to those realized by nondeterministic 2GSM's. We conclude that as opposed to the deterministic case, the two nondeterministic classes are incomparable. This is not surprising, because the nondeterminism of an MSO transduction consists of one global choice between finitely many possibilities, whereas a nondeterministic 2GSM makes a possibly unbounded number of consecutive local choices. We prove that the class of nondeterministic MSO definable string transductions is equal to the class of transductions obtained by composing a certain (nondeterministic) string relabeling and a deterministic transduction. Then we characterize the deterministic MSO definable string transductions as those nondeterministic MSO definable string transductions that happen to be functions.

In our final section, Section 7, we propose the Hennie machine [Hennie 1965] as a machine model that characterizes the MSO definable string transductions, not only in the deterministic, but also in the nondeterministic case (Theorems 25 and 26). A Hennie machine is an extension of the 2GSM in that it is allowed to rewrite the contents of its input tape (like a Turing machine). At the same time its behavior is restricted by requiring that it is finite-visit, i.e., that there is a fixed bound on the number of visits to each of the tape positions. Intuitively, the Hennie machine is more suited than the 2GSM to characterize the nondeterministic MSO definable transductions, as it has a way to record a global choice (on the input tape) whereas the number of local choices is bounded (by the finite-visit property). We conclude the paper by taking a close look at the string transductions that are nondeterministic MSO definable and at the same time realized by a 2GSM.

2. PRELIMINARIES

We recall some notions and results regarding strings, graphs, and their monadic second-order logic.

For an alphabet Σ, we use Σ^* (Σ^+) to denote the set of all (nonempty) strings over Σ. By $|w|$ we denote the length of the string w. The empty string (of length 0) is denoted by λ. A set of strings is also called a *language*.

We use $\circ$ to denote the composition of binary relations (note the order) $R_1 \circ R_2 = \{(w_1, w_3) \mid \text{there exists } w_2 \text{ such that } (w_1, w_2) \in R_1, (w_2, w_3) \in R_2\}$, and we extend it to classes of binary relations $F_1 \circ F_2 = \{R_1 \circ R_2 \mid R_1 \in F_1, R_2 \in F_2\}$. The inverse of a binary relation R is denoted R^{-1}.

A binary relation R is a (partial) *function*, if $(w, z_1) \in R$ and $(w, z_2) \in R$ imply $z_1 = z_2$. It is *finitary*, if the set $\{z \mid (w, z) \in R\}$ is finite for each w in the domain of R.

A binary relation R is also called a *transduction* (in particular when there is a logical or computational device which transforms input w into output z, for every $(w, z) \in R$). If R is a function, it is also called a *deterministic* transduction.

2.1 Directed Graphs

Let Σ and Γ be alphabets of node labels and edge labels, respectively. A *graph* over Σ and Γ is a triple $g = (V, E, \ell)$, where V is the finite nonempty set of nodes, $E \subseteq V \times \Gamma \times V$ the set of labeled edges, and $\ell\colon V \to \Sigma$ the node labeling. Note that graphs are nonempty. The set of all graphs over Σ and Γ is denoted by $\mathrm{GR}(\Sigma, \Gamma)$. We allow graphs that have both labeled and unlabeled nodes and edges by introducing a designated symbol $*$ to represent an "unlabel" in our specifications, but we omit this symbol from our drawings. We write $\mathrm{GR}(*, \Gamma)$ and $\mathrm{GR}(\Sigma, *)$ to distinguish the cases when all nodes are unlabeled, and all edges are unlabeled, respectively.

2.2 Logic for Graphs

For alphabets Σ and Γ, the monadic second-order logic $\mathrm{MSO}(\Sigma, \Gamma)$ expresses properties of graphs over Σ and Γ. The logical language uses both node variables $x, y, \ldots$ and node-set variables $X, Y, \ldots$.

There are four types of atomic formulas: $\mathrm{lab}_\sigma(x)$, meaning node x has label σ (with $\sigma \in \Sigma$); $\mathrm{edge}_\gamma(x, y)$, meaning there is an edge from x to y with label γ (with $\gamma \in \Gamma$); $x = y$, meaning nodes x and y are equal; and $x \in X$, meaning x is an element of X.

As usual, formulas are built from atomic formulas with the propositional connectives $\neg, \wedge, \vee, \to$, using the quantifiers $\forall$ and $\exists$ both for node variables and node-set variables.

A useful example [Thatcher and Wright 1968] of such a formula is the binary predicate $x \preceq y$ claiming the existence of a (directed) path from x to y:

$$(\forall X)[(x \in X \wedge \mathrm{closed}(X)) \to y \in X]$$

where $\mathrm{closed}(X) \equiv (\forall z_1)(\forall z_2)(z_1 \in X \wedge \mathrm{edge}(z_1, z_2) \to z_2 \in X)$, and $\mathrm{edge}(z_1, z_2) \equiv \bigvee_{\gamma \in \Gamma} \mathrm{edge}_\gamma(z_1, z_2)$. We also use $x \prec y$, where one additionally requires that $x \neq y$; for acyclic graphs this expresses the existence of a nonempty path from x to y.

Let φ be a formula of $\mathrm{MSO}(\Sigma, \Gamma)$ with set Ξ of free variables (of either type), and let $g = (V, E, \ell)$ be a graph in $\mathrm{GR}(\Sigma, \Gamma)$. Let ν be a valuation of φ, i.e., a mapping that assigns to each node variable $x \in \Xi$ an element $\nu(x)$ of V, and to each set variable $X \in \Xi$ a subset $\nu(X)$ of V. We write $g, \nu \models \varphi$ if φ is satisfied in the graph g, where the free variables of φ are valuated according to ν.

Let $\varphi(x_1, \ldots, x_m, X_1, \ldots, X_n)$ be an $\mathrm{MSO}(\Sigma, \Gamma)$ formula with free node variables x_i and free node-set variables X_j, and let $u_1, \ldots, u_m$ be nodes of graph g, and $U_1, \ldots, U_n$ sets of nodes of g. We write $g \models \varphi(u_1, \ldots, u_m, U_1, \ldots, U_n)$ whenever $g, \nu \models \varphi(x_1, \ldots, x_m, X_1, \ldots, X_n)$, where ν is the valuation with $\nu(x_i) = u_i$, $\nu(X_j) = U_j$.

Let Ξ be a finite set of variables. The set $\{0, 1\}^\Xi$ of 0, 1-assignments to elements of Ξ is finite, and may be considered as an alphabet. A Ξ-*valuated* graph over Σ and Γ is a graph in $\mathrm{GR}(\Sigma \times \{0, 1\}^\Xi, \Gamma)$, such that for every node variable x in Ξ there is a unique node of the graph of which the label $(\sigma, f) \in \Sigma \times \{0, 1\}^\Xi$ satisfies $f(x) = 1$.

Clearly, such a Ξ-valuated graph g determines a graph $g \mid \Sigma$ in $\mathrm{GR}(\Sigma, \Gamma)$, by dropping the $\{0, 1\}^\Xi$ component of its node labels, and it determines a valuation ν_g of the variables in Ξ, by taking

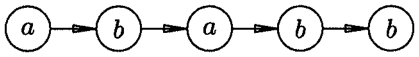
—for a node variable $x \in \Xi$, $\nu_g(x) = u$, where u is the unique node having a label (σ, f) with $f(x) = 1$,

—for a node-set variable $X \in \Xi$, $\nu_g(X) = U$, where U consists of all nodes v having a label (σ, f) with $f(X) = 1$.

For a formula φ of $\mathrm{MSO}(\Sigma, \Gamma)$ with free variables in Ξ, and a Ξ-valuated graph g we write $g \models \varphi$ if φ is true for the underlying graph under the implicitly defined valuation, i.e., if $g|\Sigma, \nu_g \models \varphi$; φ defines the set of graphs $G(\varphi) = \{g \in \mathrm{GR}(\Sigma \times \{0, 1\}^\Xi, \Gamma) \mid g \models \varphi\}$. A set of graphs is MSO *definable* if there exists a closed MSO formula that defines it.

2.3 String Representation

A string $w \in \Sigma^+$ of length $k > 0$ can be represented by the graph nd-gr(w) in $\mathrm{GR}(\Sigma, *)$, consisting of k nodes labeled by the consecutive symbols of w, with $k - 1$ (unlabeled) edges representing the successor relation for the positions of the string. In the figure below, we show nd-gr$(ababb)$. Note that for the empty string λ, nd-gr(λ) is undefined. With this representation, a formula φ of $\mathrm{MSO}(\Sigma, *)$ defines the language $L(\varphi) = \{w \in (\Sigma \times \{0, 1\}^\Xi)^+ \mid \text{nd-gr}(w) \models \varphi\}$, where Ξ is the set of free variables of φ; note that nd-gr(w) is a Ξ-valuated graph over Σ and $*$.



Given the close connection between the positions and their successor relation in a string w on the one hand, and the nodes and their connecting edges in nd-gr(w) on the other, we say that a string w satisfies a formula φ if nd-gr$(w) \models \varphi$.

The languages definable by monadic second-order formulas are exactly the regular languages, as shown by Büchi, Elgot and Trakhtenbrot.

PROPOSITION 1 [BÜCHI 1960; ELGOT 1961; TRAKHTENBROT 1962].

(1) For every formula φ of $\mathrm{MSO}(\Sigma, *)$*, $L(\varphi)$ is a regular language.*

(2) A language $K \subseteq \Sigma^+$ is regular iff there is a closed formula φ of $\mathrm{MSO}(\Sigma, *)$ *such that $K = L(\varphi)$.*

Observe that the set of all (nonempty) strings over a fixed alphabet Σ forms an MSO definable set of graphs via the above representation. The defining formula for the set $\{\text{nd-gr}(w) \mid w \in \Sigma^+\}$ over $\mathrm{MSO}(\Sigma, *)$ demands that the path relation $\preceq$ is a linear order and that the edge relation is functional. As a consequence, the set of graphs representing a language K, $\{\text{nd-gr}(w) \mid w \in K\}$, is MSO definable for every regular language K, $\lambda \notin K$.

3. TWO-WAY MACHINES

We present our model of two-way generalized sequential machines (2GSM), or two-way finite-state transducers. In order to facilitate the proof of the equivalence of two-way finite-state transductions and logically definable transductions we extend the basic model to a machine model that has its tests as well as moves (on the input tape) specified by MSO formulas. We prove the equivalence of this extended model to the basic model. An important tool in this proof is the observation that a two-way automaton is able to keep track of the state of another (one-way) finite-state automaton (proved in Lemma 3 of Hopcroft and Ullman [1967]; see also Aho et al. [1969, p. 212]). We formalize this fact by extending the 2GSM with the feature of "regular look-around." The equivalence of this model with the basic model is then proved using the related result of Chytil and Jákl [1977] stating that deterministic two-way finite-state transductions are closed under composition. The equivalence of the regular look-around model with the MSO formula model is proved using the result of Büchi, Elgot and Trakhtenbrot (Proposition 1).

Since we need several types of two-way machines, we first introduce a generic model, and then instantiate it in several ways.

A *two-way machine* (2M) is a finite-state device equipped with a two-way input tape (read only), and a one-way output tape. In each step of a computation the machine reads an input symbol, changes its internal state, outputs a string, and moves its input head, all depending on the symbol read and the original internal state.

We specify a 2M as a construct $\mathcal{M} = (Q, \Sigma_1, \Sigma_2, \delta, q_{in}, q_f)$, where Q is the finite set of states, Σ_1 and Σ_2 are the input alphabet and output alphabet, q_{in} and q_f are the initial and the final state, and δ is a finite set of *instructions*. Each instruction is of the form $(p, t,\ q, \alpha, \mu)$, where $p \in Q - \{q_f\}$ is the present state of the machine, t is a test to be performed on the input, and the triple (q, α, μ) describes the action of the machine assuming the test t is satisfied: $q \in Q$ is the new state; $\alpha \in \Sigma_2^*$ is the string written on the output tape; and μ describes the (deterministic) move of the reading head on the input tape. The precise form of these instructions varies from one model to another, in particular the form of the test t and the move μ.

The string on the input tape is marked by two special symbols, the endmarkers $\vdash$ and $\dashv$, indicating the left and right boundaries of the tape respectively. So, when processing the input string $\sigma_1 \cdots \sigma_n$, $\sigma_i \in \Sigma_1$, the tape has $n+2$ reachable positions $0, 1, \ldots, n, n+1$, containing the string $\vdash\sigma_1 \cdots \sigma_n\dashv$. The reading head is on one of these positions.

The 2M $\mathcal{M}$ realizes the transduction $m \subseteq \Sigma_1^* \times \Sigma_2^*$, such that $(w, z) \in m$ whenever there exists a computation with $\vdash w \dashv$ on the input tape, starting in initial state q_{in} with the input head on position 0 (where the symbol $\vdash$ is stored), and ending in the accepting state q_f, while z has been written on the output tape.

A 2M is *deterministic* if for all distinct instructions $(p, t_1,\ q_1, \alpha_1, \mu_1)$ and $(p, t_2,$ $q_2, \alpha_2, \mu_2)$ starting in the same state p, the tests t_1 and t_2 are mutually exclusive, i.e., for each input string w and each input position u, t_1 and t_2 are not both satisfied for the input tape $\vdash w \dashv$ with the reading head on position u. Note that the transduction m realized by a deterministic 2M $\mathcal{M}$ is a function m: $\Sigma_1^* \to \Sigma_2^*$ because the μ in the instructions describes deterministic moves of the reading head, and because there are no instructions starting in state q_f.

Without loss of generality we assume that the 2M's we consider never write more than one symbol at a time, i.e., for each instruction $(p, \sigma,\ q, \alpha, \mu)$ we have $|\alpha| \leq 1$.

We now consider the usual two-way *generalized sequential* machine (2GSM), introduced in Aho and Ullman [1970], and two new instantiations of the generic 2M model, the 2GSM with *regular look-around*, and the 2GSM with MSO-*instructions*.

3.1 2GSM

For the basic 2GSM model each instruction $(p, t,\ q, \alpha, \mu)$ in δ has $t \in \Sigma_1 \cup \{\vdash, \dashv\}$, and $\mu \in \{-1, 0, +1\}$.

A test $\sigma \in \Sigma_1 \cup \{\vdash, \dashv\}$ is satisfied if σ is the symbol under the reading head. Executing an instruction $(p, \sigma,\ q, \alpha, \epsilon) \in \delta$ the 2GSM, assuming it is in internal state p, and assuming it is reading σ on its input tape, changes its state to q, writes α to its output tape, and moves its head from the present position i to the position $i + \epsilon$ (provided $0 \leq i + \epsilon \leq n + 1$).

Note that our generic definition of determinism indeed corresponds with the usual one: no two distinct instructions starting in the same state may have the same input symbol (test).

Example 1. Consider the string transduction

$$\left\{(a^{i_1}ba^{i_2}b\cdots a^{i_n}ba^{i_{n+1}}, a^{i_1}b^{i_1}a^{i_2}b^{i_2}\cdots a^{i_n}b^{i_n}a^{i_{n+1}}) \,\middle|\, n \geq 0, i_1, \ldots, i_{n+1} \geq 0\right\}.$$

An obvious deterministic 2GSM reads each segment of a's from left to right while copying it to the output. When encountering a b it rereads the segment from right to left. This second pass writes b's to the output tape. Then, in a third and final pass, it moves right again, looking for the next segment of a's.

This machine can be implemented by taking $\Sigma_1 = \Sigma_2 = \{a, b\}$, $Q = \{1, 2, 3, 4\}$, $q_{in} = 3$, $q_f = 4$, and δ consisting of the instructions

$$\begin{array}{lll} (1, a,\ 1, a, +1), & (1, b,\ 2, \lambda, -1), & (1, \dashv,\ 4, \lambda, 0), \\ (2, a,\ 2, b, -1), & (2, \sigma,\ 3, \lambda, +1), \sigma = \vdash, b, & \\ (3, a,\ 3, \lambda, +1), & (3, \sigma,\ 1, \lambda, +1), \sigma = \vdash, b. & \end{array}$$

We have chosen $q_{in} = 3$ so that the left end marker $\vdash$ acts similarly to a b at the beginning of a segment of a's. The computation of the 2GSM on input $aaabbaba$ can be visualized as in Figure 1, where we have labeled the edges of the computation by the strings that are written to the output (with λ omitted, for convenience).

3.2 Look-Around

A 2GSM *with regular look-around* (2GSM-RLA) extends the basic 2GSM model, by allowing more complicated tests. In an instruction $(p, t,\ q, \alpha, \epsilon) \in \delta$ all components are as before for the 2GSM, except the test t, which does not consist of a single symbol σ, but of a triple $t = (R_\ell, \sigma, R_r)$, where $\sigma \in (\Sigma_1 \cup \{\vdash, \dashv\})$, and R_ℓ, R_r are regular languages such that $R_\ell, R_r \subseteq (\Sigma_1 \cup \{\vdash, \dashv\})^*$. This test t is

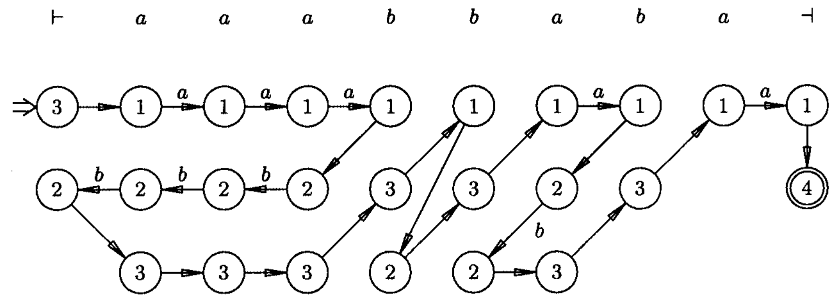


Fig. 1. Computation for (a^3b^2aba, a^3b^3aba) of the 2GSM from Example 1.

satisfied if σ is the symbol under the reading head, and the strings to the left and the right of the head belong to R_ℓ and R_r respectively.

Obviously, it suffices to have tests (R_ℓ, σ, R_r) such that $R_\ell \cdot \sigma \cdot R_r \subseteq \vdash\Sigma_1^*\dashv$. For a given 2GSM-RLA, an equivalent 2GSM-RLA with that property is obtained by changing each test (R_ℓ, σ, R_r) into (R'_ℓ, σ, R'_r) where $R'_\ell = R_\ell \cap \vdash\Sigma_1^*$ (with the exception that $R'_\ell = \{\lambda\}$ when $\sigma = \vdash$), and similarly for R'_r. We observe here that this notion of "regular look-around" generalizes the well-known notion of regular look-ahead for one-way automata (see e.g., Nijholt [1982], and Engelfriet [1977]).

3.3 MSO Instructions

For a 2GSM *with* MSO-*instructions* (2GSM-MSO) the test and the move of each instruction are given by MSO formulas. To be precise, for $(p, t, q, \alpha, \mu) \in \delta$, t is given as a formula $\varphi(x)$ in $\mathrm{MSO}(\Sigma_1 \cup \{\vdash, \dashv\}, *)$ with one free node variable x, and the move μ is given by a functional formula $\varphi(x, y)$ in $\mathrm{MSO}(\Sigma_1 \cup \{\vdash, \dashv\}, *)$ with two free node variables x and y (see below for the meaning of "functional").

A test $t = \varphi(x)$ is evaluated for the string on the input tape with x valuated as the position taken by the reading head; more precisely, as our logic is defined for graphs, t is satisfied whenever nd-gr$(\vdash w\dashv) \models \varphi(u)$, where w is the input string, and u is the node corresponding to the position of the reading head.

The 2GSM-MSO does not move stepwise on the input tape, but it "jumps" as specified by the formula $\varphi(x, y)$, as follows. Assuming the machine is in position u, it moves to a position v for which nd-gr$(\vdash w\dashv) \models \varphi(u, v)$, where we have identified positions on the input tape with their corresponding nodes of the graph nd-gr$(\vdash w\dashv)$.

To guarantee that $\varphi(x, y)$ describes deterministic moves of the reading head, we require that the relation specified by $\varphi(x, y)$ is a function, for each input string w, i.e., for every position u there is at most one position v such that nd-gr$(\vdash w\dashv) \models \varphi(u, v)$. Note that this property is expressible in the logic, and hence it is decidable: we may use the result of Büchi, Elgot, and Trakhtenbrot (Proposition 1, which is effective) to verify that it is satisfied by every string in $\vdash\Sigma_1^*\dashv$.

Example 2. Consider again the string transduction

$$\left\{(a^{i_1}ba^{i_2}b\cdots a^{i_n}ba^{i_{n+1}}, a^{i_1}b^{i_1}a^{i_2}b^{i_2}\cdots a^{i_n}b^{i_n}a^{i_{n+1}}) \,\middle|\, n \geq 0, i_1, \ldots, i_{n+1} \geq 0\right\}.$$

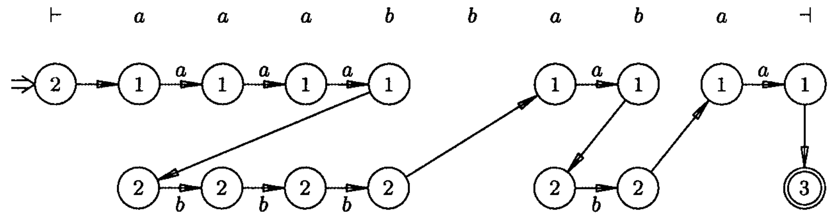


Fig. 2. Computation for (a^3b^2aba, a^3b^3aba) of the 2gsm-mso from Example 2.

It can be realized by a deterministic 2GSM-MSO $\mathcal{M}$ as follows. In state 1 it walks along a segment of a's, copying it to the output tape. Then, when the segment is followed by a b, $\mathcal{M}$ jumps back to the first a of the segment for a second pass, in state 2. When the end of the segment is reached for the second time, the machine jumps to the next segment of a's, returning to state 1. At the last a of the input the machine jumps to the right end marker, and switches to the final state 3.

We use the formulas $\text{next}_\Delta(x, y)$ to specify the first position y following x that is labeled by a symbol in Δ, and $\text{fls}_a(x, y)$ denoting the position y of the first a in the last segment of a's to the left of x. They can easily be expressed in $\text{MSO}(\Sigma \cup \{\vdash, \dashv\}, *)$.

Let $\Sigma_1 = \Sigma_2 = \{a, b\}$, $Q = \{1, 2, 3\}$, $q_{in} = 2$, $q_f = 3$, and δ consisting of

$$\begin{array}{ll}(1, \text{lab}_a(x),\ 1, a, \text{edge}_*(x, y)), & (1, \text{lab}_b(x),\ 2, \lambda, \text{fls}_a(x, y)), \\ (1, \text{lab}_\dashv(x),\ 3, \lambda, x = y), & \\ (2, \text{lab}_a(x),\ 2, b, \text{edge}_*(x, y)), & \big(2, \neg\text{lab}_a(x),\ 1, \lambda, \text{next}_{\{a,\dashv\}}(x, y)\big).\end{array}$$

The computation of the machine on input a^3b^2aba can be visualized as in Figure 2 (where, again, λ is omitted from the edges of the computation).

We abbreviate deterministic 2M's by adding a "D" to the usual abbreviation; hence we speak of 2DGSM, 2DGSM-RLA, and 2DGSM-MSO. The classes of string transductions realized by these three types of deterministic two-way machines are denoted by 2DGSM, $\text{2DGSM}^{\text{RLA}}$, and $\text{2DGSM}^{\text{MSO}}$, respectively.

Unlike their nondeterministic counterparts [Kiel 1975] (see also Lemma 15 and the discussion following Corollary 16), deterministic 2GSM's are closed under composition, as was demonstrated by Chytil and Jákl [1977]. As an essential part of the proof the fact is used (proved in Hopcroft and Ullman [1967]) that a 2DGSM can keep track of the state of another (deterministic) one-way finite-state automaton working on the same tape (from left to right or from right to left). For the left-to-right case, it is clear how to do this as long as the reading head moves to the right. Backtracking ("undoing" a move) on the occasion of a step to the left needs a rather ingenious back and forth simulation of the automaton.

PROPOSITION 2 [CHYTIL AND JÁKL 1977]. 2DGSM *is closed under composition.*

In the remainder of this section we show that the three types of deterministic machines defined above are all equivalent, i.e., that $\text{2DGSM} = \text{2DGSM}^{\text{RLA}} = \text{2DGSM}^{\text{MSO}}$.

Every 2GSM is of course a simple 2GSM-RLA, using trivial look-around tests, i.e., tests of the form (R_ℓ, σ, R_r), with $R_\ell = \vdash\Sigma_1^*$, and $R_r = \Sigma_1^*\dashv$ (with the exceptions $R_\ell = \{\lambda\}$ when $\sigma = \vdash$, and $R_r = \{\lambda\}$ when $\sigma = \dashv$).

It follows from the result of Büchi, Elgot, and Trakhtenbrot, Proposition 1, that any 2GSM-RLA can be reinterpreted as a 2GSM-MSO by changing the specification of the tests and moves into formulas, as follows.

First, consider a look-around test $t = (R_\ell, \sigma, R_r)$. As observed before, we may assume that $R_\ell \subseteq \vdash\Sigma_1^*$ (or $R_\ell = \{\lambda\}$ if $\sigma = \vdash$). Let $\psi_\ell(x)$ be a formula expressing that the string to the left of position x belongs to the regular language R_ℓ. It can be obtained from a closed formula ψ defining R_ℓ by relativizing ψ to the positions to the left of x, i.e., to $y \prec x$. (Again we have an exception for $\sigma = \vdash$. As we do not allow empty graphs, we cannot test for $R_\ell = \{\lambda\}$. In that case, set $\psi_\ell(x) \equiv$ true.) Similarly, we obtain a formula $\psi_r(x)$ expressing that the string to the right of position x belongs to the regular language R_r. Clearly, the test t is equivalent to the formula $\psi_\ell(x) \wedge \mathrm{lab}_\sigma(x) \wedge \psi_r(x)$.

Finally, one-step moves are easily translated into formulas. A move $\epsilon = +1$ is equivalent to stating that the new position is next to the original: $\mathrm{edge}_*(x, y)$. Of course, $\epsilon = -1$ is symmetric, whereas $\epsilon = 0$ is expressed by $x = y$. Note that these formulas are functional.

These observations prove the first relations between the classes of transductions.

LEMMA 3. $2\mathrm{DGSM} \subseteq 2\mathrm{DGSM}^{\mathrm{RLA}} \subseteq 2\mathrm{DGSM}^{\mathrm{MSO}}$.

The feature of 2DGSM's that they can keep track of the state of a one-way finite-state automaton (cf. the remark before Proposition 2), is modeled by us as regular look-around. Thus, for readers familiar with this feature it should be quite obvious that $2\mathrm{DGSM}^{\mathrm{RLA}} \subseteq 2\mathrm{DGSM}$. Here we prove it using Proposition 2.

LEMMA 4. $2\mathrm{DGSM}^{\mathrm{RLA}} \subseteq 2\mathrm{DGSM}$.

PROOF. By Proposition 2, 2DGSM is closed under composition. We prove the lemma by decomposing a given 2DGSM-RLA $\mathcal{M}$ into a series of 2DGSM's, together realizing the transduction of $\mathcal{M}$.

The final 2DGSM performs the required transduction, whereas all the other transductions "preprocess the tape," by adding to the original input the outcome of the various tests of $\mathcal{M}$. As we also need this information for the positions containing the end markers $\vdash$ and $\dashv$, we start by a transduction that maps input w to the string $\triangleright w \triangleleft$, where $\triangleright$ and $\triangleleft$ are new symbols. Information concerning the end positions of the tape is added to these new symbols. The other machines may ignore $\vdash$ and $\dashv$, and treat $\triangleright$ and $\triangleleft$ as if they where these end markers.

For each look-around test $t = (R_\ell, \sigma, R_r)$ of $\mathcal{M}$ we introduce a 2DGSM $\mathcal{M}_t$ that copies the input, while adding to each position the outcome of the test t for that position in the original string (ignoring any other additional information a previous transduction has added to the string). The machine $\mathcal{M}_t$ itself can be seen as the work of three consecutive 2DGSM's. The first one, simulating a deterministic finite automaton recognizing R_ℓ, checks on each position whether the prefix read belongs to R_ℓ. It adds this information to the symbol at that position. The second transducer, processing the input from right to left, simulating a deterministic finite automaton for the mirror image of R_r, adds to each position information concerning the suffix. Note that the input has been reversed in the process. This can be undone by another reversal performed by a third 2DGSM.

Once the value of each look-around test of $\mathcal{M}$ is added to the symbols of the original input string, obviously the transduction of $\mathcal{M}$ can be simulated by an ordinary 2DGSM. $\square$

The result of Büchi, Elgot, and Trakhtenbrot (Proposition 1) allows us to show that the 2GSM-MSO can be simulated by the 2GSM-RLA. Additionally we need the following (folklore) result on the structure of certain regular languages (cf. Pixton [1996, Lemma 8.1]).

LEMMA 5. *Let $\Delta \subseteq \Sigma$ be alphabets, and let $R \subseteq \Sigma^*$ be a regular language such that each string of R contains exactly one occurrence of a symbol from Δ. Then we may write R as a finite union of disjoint languages $R_\ell \cdot a \cdot R_r$, where $a \in \Delta$, and $R_\ell, R_r \subseteq (\Sigma - \Delta)^*$ are regular languages.*

PROOF. Let $\mathcal{A}$ be a deterministic finite automaton accepting R. Every path (in the state transition diagram of $\mathcal{A}$) from the initial state to a final state passes exactly one transition labeled by a symbol from Δ. For any such transition (p, a, q) of $\mathcal{A}$ let R_ℓ consist of all strings that label a path starting in the initial state of $\mathcal{A}$ and ending in p, and symmetrically, let R_r consist of all strings that label a path from q to one of the final states of $\mathcal{A}$. Obviously, R_ℓ and R_r are regular, and R is the union of the languages $R_\ell \cdot a \cdot R_r$ taken over all such transitions. Since $\mathcal{A}$ is deterministic, these languages are easily seen to be disjoint. $\square$

It is interesting to note that (as observed by one of the referees) there is a corresponding result that works entirely on the level of MSO formulas. A formula $\varphi(x)$ can be decomposed into $2k$ formulas $\varphi_{\ell,1}, \ldots, \varphi_{\ell,k}$ and $\varphi_{r,1}, \ldots, \varphi_{r,k}$ such that for every string w and every node u of nd-gr(w), nd-gr(w) $\models \varphi(u)$ iff for some j both nd-gr($w_\ell a$) $\models \varphi_{\ell,j}$ and nd-gr($a w_r$) $\models \varphi_{r,j}$, where $w = w_\ell a w_r$ and u is the node of nd-gr(w) corresponding to this occurrence of a. This fact can be proved with the help of Ehrenfeucht-Fraïssé games, e.g., see Thomas [1997a, Theorem 5]. It could equally well be used in the proof of the next lemma.

LEMMA 6. $2\text{DGSM}^{\text{MSO}} \subseteq 2\text{DGSM}^{\text{RLA}}$.

PROOF. We show how to simulate the instructions of a 2GSM-MSO by a 2GSM-RLA. Recall that such an instruction is specified as $(p, t,\ q, \alpha, \mu)$, where t is a formula $\varphi(x)$ with one free node variable, and the move μ is a (functional) formula $\varphi(x, y)$ with two free node variables.

Tests: unary node predicates. Consider a test $\varphi(x)$ in $\text{MSO}(\Sigma_1 \cup \{\vdash, \dashv\}, *)$. It can easily be simulated by regular look-around tests. Identifying $(\Sigma_1 \cup \{\vdash, \dashv\}) \times \{0, 1\}^{\{x\}}$ with $(\Sigma_1 \cup \{\vdash, \dashv\}) \times \{0, 1\}$, consider the language $L(\varphi)$, which is regular by Proposition 1. As each string of this language contains exactly one symbol with 1 as its second component, it can be written as a finite union of disjoint languages $R_\ell \cdot (\sigma, 1) \cdot R_r$, with regular languages $R_\ell, R_r \subseteq ((\Sigma_1 \cup \{\vdash, \dashv\}) \times \{0\})^*$, and

$\sigma \in \Sigma_1 \cup \{\vdash, \dashv\}$; see Lemma 5. This implies that the test $\varphi(x)$ can be simulated by a finite disjunction of the look-around tests (R'_ℓ, σ, R'_r), where each R'_ℓ, R'_r is obtained from the corresponding R_ℓ, R_r by dropping the second component (the 0-part) of the symbols. Of course, each of the alternatives (R'_ℓ, σ, R'_r) can be put as a test in a separate instruction. These tests are mutually exclusive, as the languages $R_\ell \cdot (\sigma, 1) \cdot R_r$ are disjoint.

Moves: binary node predicates. Once the test of an instruction is evaluated (to true), the move is executed, and the output is written. This move is given as a formula $\varphi(x, y)$, specifying a function from the present position x to the next position y on the input. Where the 2DGSM-MSO may "jump" to its next position, independent of the relative positions of x and y, a 2DGSM-RLA can only step to one of the neighboring positions of the tape, and has to "walk" to the next position when simulating this jump.

Before starting the excursion from x to y the 2DGSM-RLA determines its direction (left, right, or stay) by evaluating the tests $(\exists y)(y \prec x \wedge \varphi(x, y))$, $(\exists y)(x \prec y \wedge \varphi(x, y))$, and $(\exists y)(x = y \wedge \varphi(x, y))$ using the method that we have explained above. Since $\varphi(x, y)$ is functional, at most one of these tests is true.

In the sequel we assume that our target position y lies to the left of the present position x, i.e., test $(\exists y)(y \prec x \wedge \varphi(x, y))$ is true. The right-case can be treated in an analogous way; the stay-case is trivial.

Similarly to the case of tests, identify $(\Sigma_1 \cup \{\vdash, \dashv\}) \times \{0, 1\}^{\{x,y\}}$ with $(\Sigma_1 \cup \{\vdash, \dashv\}) \times \{0, 1\}^2$, and consider $L(y \prec x \wedge \varphi(x, y))$. Each string of this language contains exactly one symbol with $(0, 1)$ as its second component, the position of y, and it precedes a unique symbol with $(1, 0)$ as its second component, the position of x; all other symbols carry $(0, 0)$. It can be written as a finite disjoint union of languages $R_\ell \cdot (\tau, 0, 1) \cdot R_m \cdot (\sigma, 1, 0) \cdot R_r$, with regular languages $R_\ell, R_m, R_r \subseteq ((\Sigma_1 \cup \{\vdash, \dashv\}) \times \{(0, 0)\})^*$ and $\sigma, \tau \in \Sigma_1 \cup \{\vdash, \dashv\}$, by applying Lemma 5 twice.

Our moves are functional, meaning that there is a unique position y that satisfies the predicate $\varphi(x, y)$ with x the present position. Still before starting the excursion from x to the new position y, the 2DGSM-RLA determines which language in the union above describes this position by performing the regular look-around tests $(R'_\ell \cdot \tau \cdot R'_m, \sigma, R'_r)$, where each R'_ℓ, R'_m, R'_r is obtained from the corresponding R_ℓ, R_m, R_r by deleting the second component (the (0,0)-part) of the symbols. Note that these tests are mutually exclusive, as $\varphi(x, y)$ is functional and as the languages $R_\ell \cdot (\tau, 0, 1) \cdot R_m \cdot (\sigma, 1, 0) \cdot R_r$ are disjoint.

The 2DGSM-RLA now moves to the left. In each step it checks whether the segment of the input string between the present position (candidate y) and the starting position (corresponding to x) belongs to the regular language R'_m. This can be done by simulating a deterministic finite automaton for (the mirror image of) R'_m in the finite-state control.

Each time this segment belongs to R'_m, it performs the rla-test $(R'_\ell, \tau, \Sigma_1^* \dashv)$, to verify the requirement on the initial segment of the input. Once this last test is satisfied, it has found the position y and writes the output string. □

We summarize Lemmas 3, 4, and 6.

THEOREM 7. $2DGSM = 2DGSM^{RLA} = 2DGSM^{MSO}$.

A similar result can be obtained for nondeterministic GSM's (which realize a class of transductions we denote by 2NGSM) by the same line of reasoning. The only difference is that in Lemma 4 we need the inclusion $2DGSM \circ 2NGSM \subseteq 2NGSM$ rather than $2DGSM \circ 2DGSM \subseteq 2DGSM$ (Proposition 2). This new inclusion can be proved like the latter one [Chytil and Jákl 1977].[1]

4. DETERMINISTIC MSO DEFINABLE STRING TRANSDUCTIONS

As explained in the Preliminaries, we consider MSO logic on graphs as a means of specifying string transductions, rather than dealing directly with strings. Although we are mainly interested in graph transductions that have string-like graphs as their domain and range, occasionally we find it useful to allow more general graphs as intermediate products of our constructions.

In this section we recall the definition of (deterministic) MSO graph transductions, and from it derive the (deterministic) MSO definable string transductions. We then present some basic facts and examples.

A *deterministic* MSO *definable transduction* (studied in relation to context-free graph grammars in [1991, 1994], Engelfriet [1991a], and Engelfriet and van Oostrom [1997]) is essentially an interpretation of graphs, viewed as logical structures, with monadic second-order logic as a language (for the notion of interpretation see for example, Hodges [1993], Rabin [1977], and Seese [1992]). Thus, it is a function that yields for a given input graph a new output graph, as specified by a number of MSO formulas, one formula for each node label and edge label of the output graph (which form its signature). The formulas interpret the output graph in the input graph. Whereas in first-order interpretations the output graph is in general interpreted in a power g^n of the input graph g (see also Immerman [1999] where they are called first-order queries), MSO interpretations are one-dimensional (i.e., $n = 1$), since otherwise the translation of a formula for the output graph into one for the input graph is no longer monadic (i.e., Proposition 8, part (2) below would not be valid). But such one-dimensional interpretations are too restricted, because, in the case of strings, the output string could never be longer than the input string. For this reason, as proposed by Courcelle [1994], we allow the output graph to be interpreted in a multiple of the input graph, i.e., in a disjoint sum of n copies of the input graph. In the deterministic case that we consider here, the formulas of the interpretation are not allowed to contain parameters; the nondeterministic case (with parameters) will be considered in Section 6. To be specific, for a graph satisfying a given domain formula φ_{dom} we take copies of each of the nodes, one for each element of a finite copy set C. The label of the c-copy of node x ($c \in C$) is determined by formulas $\varphi_\sigma^c(x)$, one for each symbol σ in the output alphabet. We keep only those copies of the nodes for which exactly one of the label formulas is true. Edges are defined

[1]In order to avoid confusion later, note that we have $2DGSM \circ 2NGSM = 2NGSM$, wheras the class $2NGSM \circ 2DGSM$ strictly contains 2NGSM; see Lemma 15 and the discussion following Corollary 16.

according to formulas $\varphi_\gamma^{c_1,c_2}(x, y)$: we introduce an edge with label γ in the output graph from the c_1-copy of x to the c_2-copy of y whenever such a formula holds.

Definition 1. A *deterministic* MSO *definable (graph) transduction* τ: $\mathrm{GR}(\Sigma_1, \Gamma_1) \to \mathrm{GR}(\Sigma_2, \Gamma_2)$ is specified by

—a closed *domain formula* φ_{dom},

—a finite *copy set* C,

—*node formulas* $\varphi_\sigma^c(x)$, with one free node variable x, for every $\sigma \in \Sigma_2$ and every $c \in C$, and

—*edge formulas* $\varphi_\gamma^{c_1,c_2}(x, y)$ with two free node variables x, y, for every $\gamma \in \Gamma_2$ and all $c_1, c_2 \in C$,

where all formulas are in $\mathrm{MSO}(\Sigma_1, \Gamma_1)$.

For $g \in G(\varphi_{\mathrm{dom}})$ with node set V_g, the image $\tau(g)$ is the graph (V, E, ℓ), defined as follows. We will write u^c rather than (u, c) for elements of $V_g \times C$.

—$V = \{u^c \mid u \in V_g, c \in C$, there is exactly one $\sigma \in \Sigma_2$ such that $g \models \varphi_\sigma^c(u)\}$,

—$E = \{(u^{c_1}, \gamma, v^{c_2}) \mid u^{c_1}, v^{c_2} \in V, \gamma \in \Gamma_2, g \models \varphi_\gamma^{c_1,c_2}(u, v)\}$, and

—$\ell(u^c) = \sigma$ if $g \models \varphi_\sigma^c(u)$, for $u^c \in V$, $\sigma \in \Sigma_2$.

We will tacitly assume that V is nonempty (if necessary, adapt the domain formula). This means that $G(\varphi_{\mathrm{dom}})$ is the domain of τ.

Example 3. Let $\Sigma = \{a, b\}$. As a simple example we present an MSO definable graph transduction from $\mathrm{GR}(\Sigma, *)$ to $\mathrm{GR}(*, \{a, b, *\})$ that transforms a linear graph representing a string into a ladder, while moving the symbols from the nodes to the edges (as illustrated in Figure 3).

—Domain formula φ_{dom} expresses that the input graph is a string representation (see the end of Section 2).

—The copy set C is $\{1, 2, 3, 4\}$. Each node is copied twice, whereas the last node gets two additional copies: $\varphi_*^1 \equiv \varphi_*^2 \equiv$ true, $\varphi_*^3 \varphi_*^4 \equiv \neg(\exists y)\mathrm{edge}_*(x, y)$.

—Edges are copied with the appropriate labels, one of these in reverse: for $\sigma = a, b$, $\varphi_\sigma^{1,1} \equiv \mathrm{edge}_*(x, y) \wedge \mathrm{lab}_\sigma(x)$, $\varphi_\sigma^{2,2} \equiv \mathrm{edge}_*(y, x) \wedge \mathrm{lab}_\sigma(x)$, $\varphi_\sigma^{1,3} \equiv (x = y) \wedge \mathrm{lab}_\sigma(x)$, and $\varphi_\sigma^{4,2} \equiv (x = y) \wedge \mathrm{lab}_\sigma(x)$.

—Unlabeled steps of the ladder are introduced: $\varphi_*^{1,2} \equiv \varphi_*^{3,4} \equiv (x = y)$.

—Finally, $\varphi_\sigma^{i,j} \equiv$ false in all other cases.

The class of deterministic MSO definable graph transductions is denoted by DMSOgr. Its basic properties are summarized below, e.g., see Courcelle [1997, Proposition 5.5.6].

PROPOSITION 8.

(1) DMSOgr *is closed under composition.*

(2) The MSO *definable sets of graphs are closed under inverse transductions from* DMSOgr.

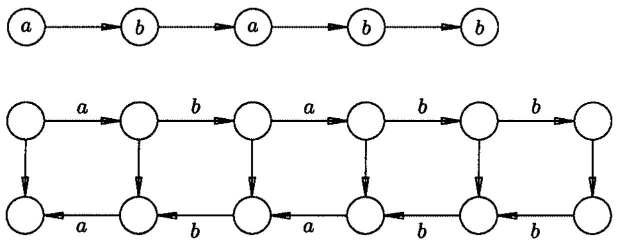
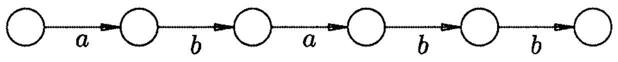


Fig. 3. Constructing a ladder, Example 3.

We now consider MSO definable graph transductions as a tool to specify string transductions.

There are two equally natural (and well-known) ways of representing a string as a graph. First, as we have seen in the Preliminaries, for a string $w \in \Sigma^*$ of length k, we may represent w by the graph nd-gr(w) in GR(Σ, $*$), consisting of k nodes labeled by the consecutive symbols of w, and $k - 1$ (unlabeled) edges representing the successor relation for the positions of the string. Recall that nd-gr(λ) is undefined. Dually, w can be represented by the graph ed-gr(w) in GR($*$, Σ), consisting of $k + 1$ (unlabeled) nodes, connected by k edges that form a path labeled by the symbols of w. In the figure below we show ed-gr($ababb$). Note that ed-gr(λ) consists of one unlabeled node.



Although nd-gr(w) more directly represents the string w in terms of its positions and their successor relation, the advantage of ed-gr(w) is that it is also defined for the empty string. Hence we prefer it for our definition of MSO definable string transductions.

Definition 2. Let Σ_1, Σ_2 be two alphabets, and let $m \subseteq \Sigma_1^* \times \Sigma_2^*$ be a string transduction. Its translation to graphs $\{(\text{ed-gr}(w), \text{ed-gr}(z)) \mid (w, z) \in m\}$ in GR($*$, Σ_1) $\times$ GR($*$, Σ_2) is denoted by ed-gr(m).

DMSOS denotes the class of all string transductions m such that ed-gr(m) belongs to DMSOgr. An element of DMSOS is called a *deterministic* MSO *definable string transduction*.

Example 4. Consider the transduction ed-gr(m), where m is the string transduction from Example 1,

$$\left\{(a^{i_1}ba^{i_2}b\cdots a^{i_n}ba^{i_{n+1}}, a^{i_1}b^{i_1}a^{i_2}b^{i_2}\cdots a^{i_n}b^{i_n}a^{i_{n+1}}) \,\middle|\, n \geq 0, i_1, \ldots, i_{n+1} \geq 0\right\}.$$

We define this transduction similar to the 2GSM-MSO from Example 2. The formulas for the construction of the output graph have nodes as their reference points, whereas the information (symbols) is attached to the edges. Hence we frequently use the formula $\text{out}_\sigma(x) \equiv (\exists y)\text{edge}_\sigma(x, y)$.

Similar to Example 2 we have an expression $\text{fps}'_a(x, y)$ denoting the first node in the present segment of a's, this time referring to outgoing edges. In the same way, we have the edge variant $\text{next}'_\Delta(x, y)$ of the original formula $\text{next}_\Delta(x, y)$.

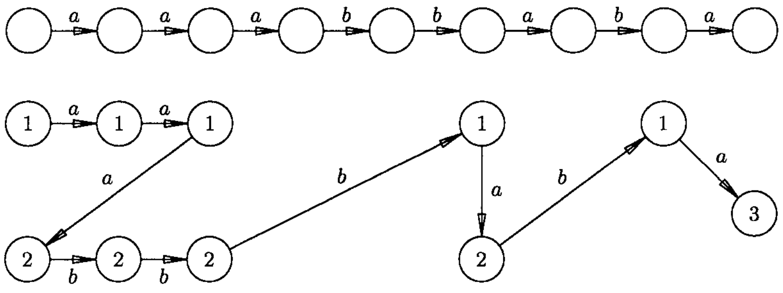


Fig. 4. Edge representation for (a^3b^2aba, a^3b^3aba), cf. Example 4.

Choosing the copy set $C = \{1, 2, 3\}$, and the domain formula defining edge representations of strings, the transduction ed-gr(m) is defined by the following formulas.

$\varphi^1_* \equiv \mathrm{out}_a(x)$
$\varphi^2_* \equiv \mathrm{out}_a(x) \wedge (\exists y)(x \preceq y \wedge \mathrm{out}_b(y))$
$\varphi^3_* \equiv \neg\mathrm{out}_a(x) \wedge \neg\mathrm{out}_b(x)$, the last node of the string,
$\varphi^{1,1}_a \equiv \mathrm{edge}_a(x, y)$
$\varphi^{1,2}_a \equiv (\exists z)(\mathrm{edge}_a(x, z) \wedge \neg\mathrm{out}_a(z)) \wedge \mathrm{fps}'_a(x, y)$
$\varphi^{1,3}_a \equiv \neg(\exists z)(\varphi^{1,1}_a(x, z) \vee \varphi^{1,2}_a(x, z))$
$\varphi^{2,2}_b \equiv \mathrm{edge}_a(x, y)$
$\varphi^{2,1}_b \equiv (\exists z)(\mathrm{edge}_a(x, z) \wedge \neg\mathrm{out}_a(z)) \wedge \mathrm{next}'_{\{a\}}(x, y)$
$\varphi^{2,3}_b \equiv \neg(\exists z)\big(\varphi^{2,1}_b(x, z) \vee \varphi^{2,2}_b(x, z)\big)$

Finally, $\varphi^{i,j}_\sigma \equiv$ false, for the remaining cases ($i = 3$).

The construction is illustrated in Figure 4 for $(a^3b^2aba, a^3b^3aba) \in m$. Note that we have put the copy numbers within the nodes.

The transition from one graph representation to the other is an MSO definable graph transduction (if we exclude the empty string). For the transduction $\mathrm{n_{to}e} = \{(\text{nd-gr}(w), \text{ed-gr}(w)) \mid w \in \Sigma^+\}$ from GR$(\Sigma, *)$ to GR$(*, \Sigma)$ this is implicit in Example 3. Its inverse transduction $\mathrm{e_{to}n} = \{(\text{ed-gr}(w), \text{nd-gr}(w)) \mid w \in \Sigma^+\}$ from GR$(*, \Sigma)$ to GR$(\Sigma, *)$ is easily seen to be MSO definable, too.

The transition from node representation to edge representation for strings does not influence the validity of the result of Büchi, Elgot, and Trakhtenbrot. In fact, we now include the empty string.

PROPOSITION 9. *A language $K \subseteq \Sigma^*$ is regular iff there is a closed formula φ of* MSO$(*, \Sigma)$ *such that* $K = \{w \in \Sigma^* \mid \text{ed-gr}(w) \models \varphi\}$.

PROOF. (If.) By Proposition 8, part (2) there is a closed formula ψ of MSO$(\Sigma, *)$ such that $G(\psi) = \mathrm{e_{to}n}(G(\varphi))$. Clearly, $L(\psi) = K - \{\lambda\}$. Hence, by Proposition 1, $K - \{\lambda\}$ is regular, so K is regular too.

The only-if direction is similar. □

We finally observe that, from Proposition 8, part (1), it immediately follows that DMSOS is closed under composition. Together with the closure under composition of 2DGSM (Proposition 2) this has been a strong indication for the equality of these two classes, proved in the next section.

5. LOGIC AND MACHINES

In this section we establish our main result, the equivalence of the MSO definable string transductions from Section 4 and the two-way finite-state transducers from Section 3, both deterministic: DMSOS = 2DGSM.

The first steps toward this result were taken already in Section 3 when we introduced the 2GSM with MSO instructions, and showed its equivalence to the basic two-way generalized sequential machine.

One technical notion that will be essential to bridge the final gap between logic and machine is modeled after Figure 1 in Example 1. That figure depicts the computation of a 2GSM on a given input string. The input string w can naturally be represented by nd-gr($\vdash w \dashv$) with nodes corresponding to positions on the input tape. On the other hand, the output string z is represented as ed-gr(z') where the edges conveniently correspond to steps of the 2GSM from one position to another (and where z is obtained from z' by erasing λ, i.e., by removing the unlabeled edges).

We introduce a notation for this representation. Let $m \subseteq \Sigma_1^* \times \Sigma_2^*$ be a string transduction. We use tape(m) to denote the graph transduction $\{$(nd-gr($\vdash w \dashv$), ed-gr(z)) $\mid (w, z) \in m\}$ in GR($\Sigma_1 \cup \{\vdash, \dashv\}, *) \times$ GR($*, \Sigma_2$).

In the definition of DMSOS we may replace the representation ed-gr(m) by tape(m), as shown in the next lemma.

LEMMA 10. *Let Σ_1 and Σ_2 be alphabets. For every string transduction m: $\Sigma_1^* \to \Sigma_2^*$,* tape($m$) $\in$ DMSOgr *iff* ed-gr(m) $\in$ DMSOgr.

PROOF. Obviously, for an alphabet Σ, the transductions tape(id) $=$ $\{$(nd-gr($\vdash w \dashv$), ed-gr(w)) $\mid w \in \Sigma^*\}$ and its inverse are MSO definable. Since DMSOgr is closed under composition (Proposition 8, part (1)), the result follows from the equalities tape(m) = tape(id) $\circ$ ed-gr(m) and ed-gr(m) = tape(id)$^{-1}$ $\circ$ tape(m). $\square$

Example 5. Consider the transduction tape(m), where m is the string transduction from Example 1,

$$\left\{(a^{i_1}ba^{i_2}b\cdots a^{i_n}ba^{i_{n+1}}, a^{i_1}b^{i_1}a^{i_2}b^{i_2}\cdots a^{i_n}b^{i_n}a^{i_{n+1}}) \,\middle|\, n \geq 0, i_1, \ldots, i_{n+1} \geq 0\right\}.$$

Previously we have shown that $m \in$ 2DGSM; here we will demonstrate that tape(m) is an MSO definable graph transduction.

Recall the predicate $\text{next}_\Delta(x, y)$ from Example 2.

For tape(m) the domain formula specifies linear graphs of the form nd-gr($\vdash w \dashv$), $w \in \{a, b\}^*$, the copy set C is $\{1, 2, 4\}$, and we have formulas

$\varphi_*^1 \equiv \text{lab}_a(x)$,
$\varphi_*^2 \equiv \text{lab}_a(x) \wedge (\exists y)(x \preceq y \wedge \text{lab}_b(y))$,
$\varphi_*^4 \equiv \text{lab}_\dashv(x)$,
$\varphi_a^{1,1} \equiv \text{edge}_*(x, y)$,
$\varphi_a^{1,2} \equiv (x = y) \wedge \neg(\exists z)(\text{edge}_*(x, z) \wedge \text{lab}_a(z))$,

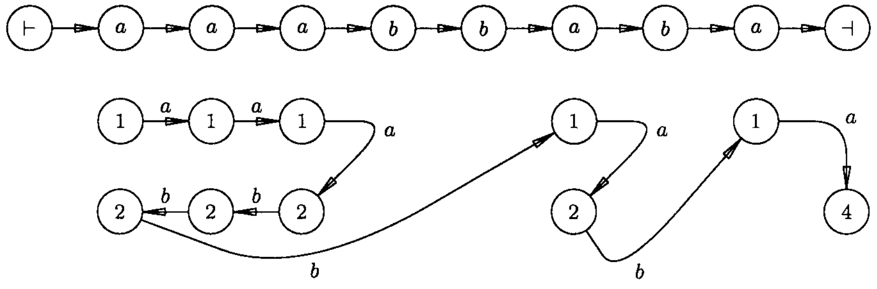


Fig. 5. MSO definable transduction tape(m) from Example 5.

$\varphi_a^{1,4} \equiv \text{edge}_*(x, y)$,
$\varphi_b^{2,2} \equiv \text{edge}_*(y, x)$,
$\varphi_b^{2,1} \equiv (\exists z)(\text{next}_{\{b\}}(x, z) \wedge \text{next}_{\{a\}}(z, y)) \wedge \neg(\exists z)(\text{edge}_*(z, x) \wedge \text{lab}_a(z))$, i.e., connect to the first a of the next segment when we are at the first a of the present segment,
$\varphi_b^{2,4} \equiv \neg(\exists z)(\varphi_b^{2,1}(x, z) \vee \varphi_b^{2,2}(x, z))$,
$\varphi_\sigma^{i,j} \equiv$ false, in all other cases.

Note that the output of the transduction (cf. the lower graph in Figure 5) is obtained by contracting unlabeled paths in the computation graph of the 2DGSM from Example 1, Figure 1.

The observation from the example is generally true: a string transduction m is realized by a 2DGSM if and only if its graph representation tape(m) is MSO definable. We prove the two implications separately.

LEMMA 11. *Let* $m : \Sigma_1^* \to \Sigma_2^*$ *be a string transduction. If* $m \in$ 2DGSM*, then* tape(m) $\in$ DMSOgr.

PROOF. Let $\mathcal{M} = (Q, \Sigma_1, \Sigma_2, \delta, q_{in}, q_f)$ be a 2DGSM realizing the string transduction m: $\Sigma_1^* \to \Sigma_2^*$, and consider a fixed input string $w = \sigma_1 \cdots \sigma_n$, $\sigma_i \in \Sigma_1$ for $i = 1, \ldots, n$. Additionally we use $\sigma_0 = \vdash$ and $\sigma_{n+1} = \dashv$.

We can visualize the "computation space" of $\mathcal{M}$ on w by constructing a graph $\gamma_{\mathcal{M}}(w)$ that has as its nodes the pairs $\langle p, i\rangle$, where p is a state of $\mathcal{M}$, and $i \in \{0, 1, \ldots, n, n+1\}$ is one of the positions of the input tape carrying $\vdash w \dashv$. The edges of $\gamma_{\mathcal{M}}(w)$ are chosen in accordance with the instruction set δ of $\mathcal{M}$: for each instruction $(p, \sigma,\ q, \alpha, \epsilon)$ in δ there is an edge from $\langle p, i\rangle$ to $\langle q, i + \epsilon\rangle$ if σ_i equals σ. The edge is labeled by the output symbol $\alpha \in \Sigma_2 \cup \{\lambda\}$. In this context we will consider λ as a labeling symbol (rather than as a string of length zero) in order to avoid notational complications.

In Figure 6 we illustrate the computation space for the 2DGSM from Example 1 on input a^3b^2aba (with output λ omitted, as usual). The computation on that input is represented as a bold path (cf. Figure 1).

As $\mathcal{M}$ is deterministic, every node of $\gamma_{\mathcal{M}}(w)$ has at most one outgoing edge. The output of the computation of $\mathcal{M}$ on w can then be read from $\gamma_{\mathcal{M}}(w)$ by starting in node $\langle q_{in}, 0\rangle$, representing $\mathcal{M}$ in its initial configuration, and following the path along the outgoing edges. The computation is successful if it

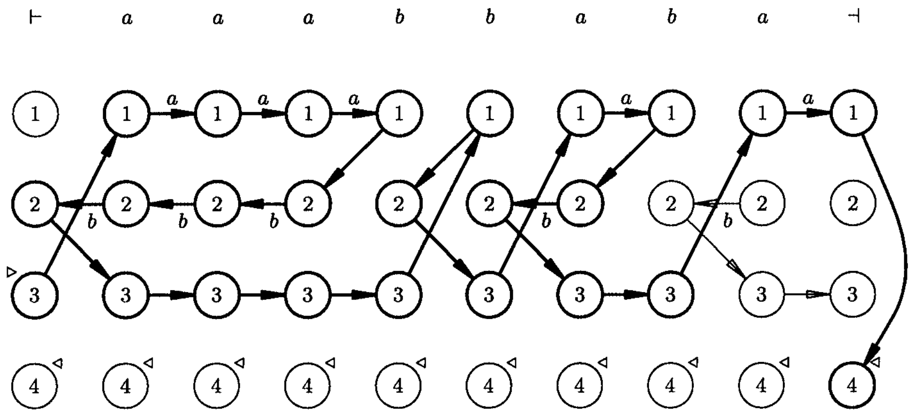


Fig. 6. Computation space $\gamma_{\mathcal{M}}(a^3b^2aba)$ for the 2DGSM $\mathcal{M}$ in Example 1.

ends in a final configuration $\langle q_f, k\rangle$. In that case the path from initial to final configuration is acyclic, and thus the ed-gr representation of the output string (with additional λ's) is a subgraph of $\gamma_{\mathcal{M}}(w)$. We will mark the initial and final nodes of $\gamma_{\mathcal{M}}(w)$ by special labels $\rhd$ and $\lhd$; the other nodes remain unlabeled (represented in our specification by $*$).

Note that the graph $\gamma_{\mathcal{M}}(w)$ does not only represent the computation of $\mathcal{M}$ on w starting in the initial state and 0th position of the tape (marked by $\vdash$) but rather all possible computations that result from placing $\mathcal{M}$ on an arbitrary position of the tape, in an arbitrary state.

We construct two MSO graph transductions, the composition of which maps nd-gr($\vdash w \dashv$) to ed-gr(z) for each $(w, z) \in m$. As DMSOgr is closed under composition (Proposition 8), this proves the lemma.

The first graph transduction τ_1 maps nd-gr($\vdash w \dashv$) to $\gamma_{\mathcal{M}}(w)$. The second graph transduction τ_2 selects the path in $\gamma_{\mathcal{M}}(w)$ corresponding to the successful computation of $\mathcal{M}$ on w (if it exists) by keeping only those nodes that are reachable from a node labeled $\rhd$ and lead to a node labeled $\lhd$. Moreover, it also discards the nodes that have an outgoing edge labeled by λ (used as a symbol representing the empty string), as it additionally contracts paths consisting of these edges.

We discuss the first transduction in some detail, while leaving the second one to the reader.

Constructing $\gamma_{\mathcal{M}}(w)$. Let τ_1: GR($\Sigma_1 \cup \{\vdash, \dashv\}, *$) $\to$ GR($\{*, \rhd, \lhd\}, \Sigma_2 \cup \{\lambda\}$) be the graph transduction that constructs $\gamma_{\mathcal{M}}(w)$. We follow the general description above, and formalize τ_1 as an MSO transduction.

The domain formula of the transduction specifies that the graph is of the form nd-gr($\vdash w \dashv$) for some string w. The copy set equals $C = Q$, where Q is the set of states of $\mathcal{M}$. The node $\langle q, i\rangle$ of $\gamma_{\mathcal{M}}(w)$ is identified with u_i^q, the q-copy of the node u_i of nd-gr($\vdash w \dashv$) corresponding to the ith position of the input tape, labeled with σ_i.

The labels of the edges are chosen according to the instructions of $\mathcal{M}$. For $\alpha \in \Sigma_2 \cup \{\lambda\}$, $p, q \in Q$, and $\epsilon \in \{-1, 0, +1\}$ let step$[\epsilon]_\alpha^{p,q}(x)$ be the disjunction of all lab$_\sigma(x)$ such that $(p, \sigma, q, \alpha, \epsilon) \in \delta$.

Then,

$$\begin{aligned}\varphi_\alpha^{p,q} \equiv\ & \big(\mathrm{edge}_*(x, y) \wedge \mathrm{step}[+1]_\alpha^{p,q}(x)\big)\\ & \vee\big(x = y \wedge \mathrm{step}[0]_\alpha^{p,q}(x)\big)\\ & \vee\big(\mathrm{edge}_*(y, x) \wedge \mathrm{step}[-1]_\alpha^{p,q}(x)\big).\end{aligned}$$

All copies of the nodes are present, with special labels for initial and final nodes:

$\varphi_\triangleright^q \equiv \mathrm{lab}_\vdash(x)$, when $q = q_{in}$, and $\varphi_\triangleright^q \equiv$ false, otherwise.
$\varphi_\triangleleft^q \equiv$ true, when $q = q_f$, and $\varphi_\triangleleft^q \equiv$ false, otherwise.
$\varphi_*^q \equiv \neg\varphi_\triangleright^q(x) \wedge \neg\varphi_\triangleleft^q(x)$.

Note that we assume that $q_{in} \neq q_f$, in order to avoid that both $\varphi_\triangleright^q$ and $\varphi_\triangleleft^q$ are defined for the initial node. □

Since we have extended the 2DGSM with the feature of MSO-instructions, cf. Theorem 7, the converse of the previous result has a rather straightforward proof.

LEMMA 12. *Let $m : \Sigma_1^* \to \Sigma_2^*$ be a string transduction. If* tape(m) $\in$ DMSOgr*, then $m \in$ 2DGSM$^{\mathrm{MSO}}$.*

PROOF. Starting with the MSO transduction tape(m): $\mathrm{GR}(\Sigma_1 \cup \{\vdash, \dashv\}, *) \to \mathrm{GR}(*, \Sigma_2)$ we build a 2DGSM-MSO $\mathcal{M}$ for m that closely follows the MSO specification of tape(m).

Assume tape(m) is specified by domain formula φ_{dom}, copy set C, node formulas φ_*^c, $c \in C$, and edge formulas $\varphi_\sigma^{c_1,c_2}$, $c_1, c_2 \in C$, $\sigma \in \Sigma_2$. The state set of $\mathcal{M}$ is equal to the copy set C with two new symbols q_{in} and q_f added as initial and final states. Now, when $\varphi_\sigma^{c_1,c_2}(u, v)$ is true for a pair u, v of nodes, then $\mathcal{M}$, visiting the position corresponding to u of the input tape in state c_1, may move to the position corresponding to v changing to state c_2, while writing σ to the output tape.

Note that, for each input graph g, tape(m)(g) defines a graph representation of a string; hence at most one of these formulas defines an edge in a given position (node u) and a given state (copy c_1). However, in general the formula $\varphi_\sigma^{c_1,c_2}(x, y)$ is only functional, and the formulas $(\exists y)(\varphi_\sigma^{c_1,c_2}(x, y))$ are only mutually exclusive, as far as graphs g satisfying the domain formula φ_{dom} are concerned, and for these graphs only when restricted to nodes for which the respective c_1 and c_2 copies are defined. Since our formal definition of 2DGSM-MSO demands functional moves and mutually exclusive tests, we consider the formulas $\psi_\sigma^{c_1,c_2}(x, y) \equiv \varphi_\sigma^{c_1,c_2}(x, y) \wedge \varphi_*^{c_1}(x) \wedge \varphi_*^{c_2}(y) \wedge \varphi_{\mathrm{dom}}$.

The instructions of $\mathcal{M}$ are of the form

$$\big(c_1, (\exists y)\big(\psi_\sigma^{c_1,c_2}(x, y)\big),\ c_2, \sigma, \psi_\sigma^{c_1,c_2}(x, y)\big).$$

If, for given c_1, none of the above tests are satisfied, the present node has no successor, which indicates the last position of the output string. In that case, $\mathcal{M}$ halts in state q_f.

Initially $\mathcal{M}$ has to find the unique node of the output graph that has no incoming edges. In the initial state q_{in} this node is found by testing all possibilities, for $c_2 \in C$:

$$\big(q_{in}, (\exists y)\big[\varphi_*^{c_2}(y) \wedge \neg\mathrm{incom}^{c_2}(y)\big],\ c_2, \lambda, \varphi_*^{c_2}(y) \wedge \neg\mathrm{incom}^{c_2}(y)\big)$$

where $\mathrm{incom}^{c_2}(y) \equiv (\exists x) \bigvee_{c_1 \in C, \sigma \in \Sigma_2} (\psi_\sigma^{c_1,c_2}(x, y))$. □

We complete the section by deriving the equivalence between the deterministic MSO definable string transductions and the deterministic two-way finite-state transductions, uniting logic and machines.

THEOREM 13. DMSOS = 2DGSM.

PROOF. By Lemma 10, $m \in$ DMSOS iff tape(m) $\in$ DMSOgr. Hence, the last two results show 2DGSM $\subseteq$ DMSOS $\subseteq$ 2DGSM$^{\mathrm{MSO}}$. Equality follows by Theorem 7. □

Since the proof of Theorem 13 is effective, it follows immediately from Gurari [1982] that the equivalence problem for deterministic MSO definable string transductions is decidable.

COROLLARY 14. *It is decidable whether or not two deterministic* MSO *definable string transductions are equal.*

6. NONDETERMINISM

In this section we study a nondeterministic variant of MSO definable graph transductions, and their derived string relatives. First, we observe that nondeterministic MSO string transductions cannot be characterized using nondeterministic 2GSM's (Corollary 16), unlike the deterministic case from the previous section. Then we investigate the relationship between nondeterministic and deterministic MSO string transductions (Theorems 19 and 21).

6.1 Nondeterministic MSO Definable Transductions

To obtain a nondeterministic variant of MSO definable transductions all the formulas of the deterministic version may now have additional free node-set variables $X_1, \ldots, X_k$, called parameters, the same for each of the formulas (cf. Hodges [1993], Rabin [1977], and Courcelle [1991; 1994]). For each valuation of the parameters (by sets of nodes of the input graph) that satisfies the domain formula, the other formulas define the output graph as before. Hence each valuation may lead to a different output graph for the given input graph: nondeterminism. This way of introducing nondeterminism into logic is quite natural (it may be viewed as a "global" existential quantification), and it seems hard to imagine another way to do it. As we will show, it does not correspond to the nondeterminism of 2GSM's, for which all choices are "local," but rather to that of Hennie machines (discussed in the next section).

Nondeterministic MSO definable graph transductions have been used, e.g., in the theory of context-free graph languages. As an easy example, there is a nondeterministic MSO definable transduction (with one parameter) that translates a graph into each of its induced subgraphs. Since the class of context-free graph languages is closed under nondeterministic MSO definable transductions (cf., Courcelle [1994]), this shows that if a set of graphs is context-free then so is the set of all its induced subgraphs.

Formally, a *nondeterministic* MSO *definable (graph) transduction* $\tau \subseteq \mathrm{GR}(\Sigma_1, \Gamma_1) \times \mathrm{GR}(\Sigma_2, \Gamma_2)$ is specified by

—a set of *parameters* $X_1, \ldots, X_k$, $k \geq 0$,
—a *domain formula* $\varphi_{\mathrm{dom}}(X_1, \ldots, X_k)$,
—a finite *copy set* C,
—*node formulas* $\varphi_\sigma^c(x, X_1, \ldots, X_k)$ for $\sigma \in \Sigma_2$, $c \in C$, and
—*edge formulas* $\varphi_\gamma^{c_1,c_2}(x, y, X_1, \ldots, X_k)$ for $\gamma \in \Gamma_2$, $c_1, c_2 \in C$,

where all formulas are in $\mathrm{MSO}(\Sigma_1, \Gamma_1)$.

Recall from Section 2 that an input graph together with a valuation of the parameters can be represented by a Ξ-valuated graph g which has node labels in $\Sigma_1 \times \{0,1\}^\Xi$ (where $\Xi = \{X_1, \ldots, X_k\}$) such that $g \mid \Sigma_1$ is the input graph, and ν_g is the valuation. By definition, $g \in G(\varphi_{\mathrm{dom}})$ iff $g \mid \Sigma_1, \nu_g \models \varphi_{\mathrm{dom}}(X_1, \ldots, X_k)$.

For each $g \in G(\varphi_{\mathrm{dom}})$ we define the graph $\hat{\tau}(g)$ similar to $\tau(g)$ in Definition 1. The nodes of $\hat{\tau}(g)$ are defined using $g \mid \Sigma_1 \models \varphi_\sigma^c(u, U_1, \ldots, U_k)$, where $U_i = \nu_g(X_i)$, rather than $g \models \varphi_\sigma^c(u)$, and similarly for the edges and node labeling of $\hat{\tau}(g)$. The transduction τ is then defined as follows: $\tau = \{(g \mid \Sigma_1, \hat{\tau}(g)) \mid g \in G(\varphi_{\mathrm{dom}})\}$.

Example 6. Let $m \subseteq \{a\}^* \times \{a, b, \#\}^*$ be the relation

$$\{(a^n, w\#w) \mid n \geq 0, w \in \{a, b\}^*, |w| = n\}.$$

The relation ed-gr(m) can be realized by a nondeterministic MSO definable transduction, with parameters X_a and X_b. The nodes of the input graph are copied twice, and the parameters determine whether the outgoing edge of a node in the input is copied as a-edge or b-edge, respectively.

The components of the transduction are as follows. The copy set equals $C = \{1, 2\}$, the domain formula $\varphi_{\mathrm{dom}}(X_a, X_b)$ expresses that the input graph is a string representation, and additionally that the sets X_a and X_b form a partition of its nodes.

All input nodes are copied twice: $\varphi_*^1(x, X_a, X_b) \equiv \varphi_*^2(x, X_a, X_b) \equiv \mathrm{true}$.

The edge labels are changed according to the sets X_a and X_b; additionally the last node of the first copy is connected to the first node of the second copy by an #-edge:

$\varphi_\sigma^{1,1}(x, y, X_a, X_b) \equiv \varphi_\sigma^{2,2}(x, y, X_a, X_b) \equiv \mathrm{edge}_a(x, y) \wedge x \in X_\sigma$, for $\sigma = a, b$,
$\varphi_\#^{1,2}(x, y, X_a, X_b) \equiv \neg(\exists z)\mathrm{edge}_a(x, z) \wedge \neg(\exists z)\mathrm{edge}_a(z, y)$,
$\varphi_\sigma^{i,j}(x, y, X_a, X_b) \equiv \mathrm{false}$, for all other combinations i, j, σ.

Mapping aaa to $abb\#abb$ can be realized by taking the valuation $\nu(X_a) = \{1\}$, $\nu(X_b) = \{2, 3, 4\}$.

Note that this example can be changed such that it uses only one parameter, as the sets represented by the parameters are complementary.

We use NMSOgr and NMSOS to denote the nondeterministic counterparts of the classes DMSOgr and DMSOS, respectively. Recall (from the end of Section 3) that 2NGSM denotes the class of (nondeterministic) 2GSM transductions.

6.2 Comparing NMSOS and 2NGSM

Unlike the deterministic case, the power of the nondeterministic 2GSM is incomparable to that of the nondeterministic MSO definable string transduction. First, because the number of parameter valuations is finite, every nondeterministic MSO transduction is finitary, i.e., it maps each input string into a finite number of output strings. This is not true for the 2GSM, which can realize the (nonfinitary) transduction $\{(a^n, a^{mn}) \mid m, n \geq 1\}$, by repeatedly copying the input string to the output, nondeterministically deciding when to terminate the repetition.

On the other hand, the nondeterministic MSO transduction of the previous example cannot be realized by a 2GSM.

LEMMA 15. *Let* $m \subseteq \{a\}^* \times \{a, b, \#\}^*$ *be the relation* $\{(a^n, w\#w) \mid n \geq 0, w \in \{a, b\}^*, |w| = n\}$. *Then* $m \notin$ 2NGSM.

PROOF. Assume m is realized by a (nondeterministic) 2GSM $\mathcal{M}$ with k states. Choose n such that $2^n > k \cdot (n+2)$. Consider the behavior of $\mathcal{M}$ on input a^n. The input tape, containing $\vdash a^n \dashv$, has $n+2$ positions. Hence, $\mathcal{M}$ has $k \cdot (n+2)$ configurations on this input. Consider the configuration assumed by $\mathcal{M}$ when it has just written the symbol # on its output tape. As there are 2^n possible output strings $w\#w$ for a^n, there exist two strings w_1 and w_2 for which this configuration is the same. This means that we can switch the computation of $(a^n, w_1\#w_1)$ halfway to the computation of $(a^n, w_2\#w_2)$ obtaining a computation for $(a^n, w_1\#w_2)$ with $w_1 \neq w_2$, which is not an element of m. $\square$

COROLLARY 16. NMSOS *and* 2NGSM *are incomparable.*

It is not difficult to see that the relation m from Lemma 15 can be realized by the composition of two 2GSM's, the first nondeterministically mapping a^n to a string $w \in \{a, b\}^*$ with $|w| = n$, the second (deterministically) doubling its input w to $w\#w$. This shows that 2NGSM is not closed under composition, as proved in Kiel [1975] for the corresponding classes of output languages.[2] It even shows that 2NGSM $\circ$ 2DGSM strictly contains 2NGSM, whereas 2DGSM $\circ$ 2NGSM = 2NGSM, cf. Section 3. However, the nondeterministic MSO transductions are closed under composition [Courcelle 1997, Proposition 5.5.6].

PROPOSITION 17. NMSOgr*, and consequently* NMSOS*, is closed under composition.*

6.3 Relating NMSOS and DMSOS

By RELgr we denote the class of (nondeterministic) node relabelings for graphs. A relation in $\mathrm{GR}(\Sigma_1, \Gamma) \times \mathrm{GR}(\Sigma_2, \Gamma)$ is a *node relabeling* if there exists a relation $R \subseteq \Sigma_1 \times \Sigma_2$ such that the images of a graph g are exactly those graphs that can be obtained from g by replacing every occurrence of a node label σ by an element of $R(\sigma)$, leaving edges and their labels unchanged.

[2]In fact, the classes 2NGSM^k of compositions of k 2GSM transductions form a strict hierarchy, as proved in Greibach [1978a], and Engelfriet 1982, 1991b (again for the corresponding classes of output languages).

We use REL to denote the class of (nondeterministic) *string relabelings*, related to RELgr through the mapping nd-gr.

We observe the following elementary relationship between deterministic and nondeterministic MSO definable graph transductions.

THEOREM 18. NMSOgr = RELgr $\circ$ DMSOgr.

PROOF. The proof of the first inclusion NMSOgr $\subseteq$ RELgr $\circ$ DMSOgr is implicit in our definition of NMSOgr. The nondeterminism of an MSO transduction τ with parameters $X_1, \ldots, X_k$ can be "preprocessed" by a node relabeling ρ that maps each node label $\sigma \in \Sigma_1$ nondeterministically to a symbol $(\sigma, f) \in \Sigma_1 \times \{0, 1\}^{\Xi}$, where $\Xi = \{X_1, \ldots, X_k\}$. Thus, ρ transforms the input graph into a Ξ-valuated graph. It is straightforward to change the formulas that specify τ into formulas for a deterministic MSO transduction $\hat{\tau}$ such that $\tau = \rho \circ \hat{\tau}$: each subformula $x \in X_i$ is changed into the disjunction of all $\mathrm{lab}_{(\sigma,f)}(x)$ with $f(X_i) = 1$.

For the converse inclusion NMSOgr $\supseteq$ RELgr $\circ$ DMSOgr, it suffices to note that each node relabeling is a nondeterministic MSO definable graph transduction. The inclusion then follows from the closure of NMSOgr under composition, Proposition 17.

Let $R \subseteq \Sigma_1 \times \Sigma_2$ define a node relabeling. We formalize it as MSO graph transduction from $\mathrm{GR}(\Sigma_1, \Gamma)$ to $\mathrm{GR}(\Sigma_2, \Gamma)$ by choosing parameters X_τ, $\tau \in \Sigma_2$, with the intended meaning that a node belonging to X_τ will be relabeled into τ.

The domain formula φ_{dom} expresses that the X_τ form an "admissible" parameter set by demanding each node to be in exactly one of the X_τ, and additionally, if a node has label σ, then X_τ containing this node satisfies $\tau \in R(\sigma)$. Each node is copied once ($C = \{1\}$), relabeled according to X_τ: $\varphi_\tau^1 \equiv x \in X_\tau$. $\square$

As we have observed, any string relabeling can be "lifted" to a graph node relabeling using the graph interpretation nd-gr of strings. By restricting the previous result to those graph transductions that result from strings, we obtain a result for MSO definable string transductions, relating the deterministic and undeterministic variants by means of a string relabeling. Unfortunately, this immediate consequence is valid for the node interpretation of strings only as the definition of NMSOgr is based upon node set variables. The translation to the edge representation needs some additional care, as we have to avoid the empty string.

In addition to REL, we need MREL denoting the class of *marked string relabellings*, that map a string w first to the "marked version" $\vdash w \dashv$, and then apply a string relabelling.

THEOREM 19. NMSOS = MREL $\circ$ DMSOS.

PROOF. Let $m \subseteq \Sigma_1^* \times \Sigma_2^*$ be a string transduction. Recall from Section 5 that $\mathrm{tape}(m) = \{(\text{nd-gr}(\vdash w \dashv), \text{ed-gr}(z)) \mid (w, z) \in m\}$. We show that the following statements are equivalent.

(1) $m \in$ NMSOS

(2) tape(m) $\in$ NMSOgr

(3) $m \in$ MREL $\circ$ DMSOS

(1) iff (2). By definition, $m \in$ NMSOS iff ed-gr(w) $\in$ NMSOgr. Now observe that this equivalence is the nondeterministic analogue of Lemma 10, and can be proved in the same way (using Proposition 17 instead of Proposition 8).

(2) implies (3). By Theorem 18, tape(m) $\in$ RELgr $\circ$ DMSOgr. Hence, tape(m) can be decomposed into a node relabeling τ_1 and a deterministic MSO definable graph transduction τ_2. By definition, τ_1 maps any input graph nd-gr($\vdash w \dashv$) to a (nonempty) graph nd-gr(w'). The second transduction τ_2 then maps nd-gr(w') to ed-gr(z) for z such that $(w, z) \in m$. We may assume that the range of τ_2 contains graph representations of strings only, with respect to ed-gr.

Again by definition, τ_1 defines a string relabeling (mapping the string $\vdash w \dashv$ to the string w'), and consequently it determines a marked relabeling m_1 (mapping w to w'). Thus, $m_1 = \{(w, w') \mid (\text{nd-gr}(\vdash w \dashv), \text{nd-gr}(w')) \in \tau_1\}$.

Consider $\mathrm{e_{to}n} \circ \tau_2$, mapping ed-gr($w'$) to ed-gr($z$) when (nd-gr($w'$), ed-gr($z$)) $\in \tau_2$. As both $\mathrm{e_{to}n}$ and τ_2 belong to DMSOgr, so does their composition, and consequently the string transduction $m_2 = \{(w', z) \mid (\text{nd-gr}(w'), \text{ed-gr}(z)) \in \tau_2\}$ is in DMSOS (because ed-gr(m_2) $= \mathrm{e_{to}n} \circ \tau_2$, by our previous assumption on τ_2).

We conclude that $m = m_1 \circ m_2$ is in MREL $\circ$ DMSOS.

(3) implies (1). It is an easy exercise to show that a marked relabeling belongs to DMSOS, along the lines of the second part of the proof of Theorem 18, this time referring to outgoing edge labels, rather than node labels (and copying the end nodes twice). Then MREL $\circ$ DMSOS $\subseteq$ NMSOS $\circ$ NMSOS $\subseteq$ NMSOS by Proposition 17. □

For completeness we note that the above result cannot be strengthened to NMSOS = REL $\circ$ DMSOS, as the relations on the right-hand side are functional for the empty string λ. This is not necessarily true for NMSOS, as shown in the next example.

Example 7. The string transduction $\{(\lambda, a), (\lambda, b)\}$ in $a^* \times \{a, b\}^*$ can clearly be realized by a nondeterministic MSO transduction. The (unique) input node is copied twice, and the single parameter X determines whether λ is mapped to a or to b: $\varphi_a^{1,2} \equiv x \in X$, $\varphi_b^{1,2} \equiv x \notin X$.

Combining the previous result (that relates the nondeterministic and deterministic MSO transductions) with the correspondence between deterministic MSO transductions and deterministic GSM mappings of Theorem 13, we directly obtain the following result.

THEOREM 20. NMSOS = MREL $\circ$ 2DGSM.

In Theorem 19 we have seen a decomposition result for NMSOS, relating the nondeterministic and deterministic variants. It turns out that DMSOS can be characterized within NMSOS using a behavioral property of the transductions.

THEOREM 21. *Let m be a string transduction. Then $m \in$* DMSOS *iff $m \in$* NMSOS *and m is a function.*

PROOF. By definition, every deterministic MSO definable string transduction is a function, hence the implication from left to right.

For the implication from right to left it suffices to show that for every nondeterministic MSO string transduction m there exists a deterministic MSO string transduction $m' \subseteq m$ with the same domain as m. Such an m' can be obtained from m by existentially quantifying the parameters $X_1, \ldots, X_n$ in all the formulas of m, and requiring that they are the least node-sets that satisfy the domain formula, in a certain linear order.

Two sets of nodes X and Y can be compared lexicographically, by viewing them as bit strings, using the linear ordering of the nodes in the input string:

$$X < Y \equiv (\exists x)(x \notin X \wedge x \in Y \wedge (\forall y)[y \prec x \to (y \in X \leftrightarrow y \in Y)]).$$

Then sequences of sets $X_1, \ldots, X_n$ can be compared lexicographically in the usual way. □

This characterization is similar to the one of 2DGSM (=DMSOS by Theorem 13) within 2NGSM, of which the proof is given (or, rather, sketched) in Engelfriet [1982, Theorem 4.9]. For completeness, we give a proof of this result within the frame of the present paper.

THEOREM 22. *Let m be a string transduction. Then $m \in$ 2DGSM iff $m \in$ 2NGSM and m is a function.*

PROOF. The implication from left to right is obvious. It remains to show that if m is a function in 2NGSM, then it belongs to NMSOS. Then we apply Theorem 21 to obtain $m \in$ DMSOS = 2DGSM.

Note that Lemma 10 also holds in the nondeterministic case, as NMSOgr is closed under composition. Thus, we may prove that $m \in$ NMSOS by showing that tape(m) $\in$ NMSOgr, in the fashion of Lemma 11.

So, assume the nondeterministic GSM $\mathcal{M}$ realizes a function m. Consider a computation of $\mathcal{M}$ for $(w, z) \in m$. We may assume that it never enters the same state twice in the same position of the tape. Otherwise that computation on input w would contain a loop (a segment of the computation starting and ending in the same tape position in the same state). Such a loop cannot produce any output, because m is a function, and can be omitted from the computation without changing the output.

Now reconsider the proof of Lemma 11. We apply the same construction, except that in τ_2 we use one parameter X such that the subgraph $\gamma_{\mathcal{M}}$ induced by the nodes in X is a string representation, from initial to final configuration. □

7. HENNIE MACHINES

In this section we propose a machine model that is suited to capture the notion of MSO definable string transductions, both in the deterministic and the nondeterministic case, unlike the 2GSM, a model which worked well only in the deterministic case (Theorem 13, Corollary 16). In the previous section we have seen two examples of string transductions that mark the incomparability between the two classes NMSOS and 2NGSM.

On the one hand, nondeterministic MSO definable transductions are finitary, due to the finite number of possible global choices of the parameter valuation. There is no such restriction for the 2GSM which may make an unbounded number of local choices, to realize the string transduction $\{(a^n, a^{mn}) \mid m, n \geq 1\}$.

On the other hand, the 2GSM is able to "memorize" only a finite number of the local choices it has made to use them later in the computation. For a nondeterministic MSO definable transduction a linear number of choices globally fixed in the parameters, may be used several times as exemplified by the transduction mapping a^n to $w\#w$ with $w \in \{a, b\}^*$, $|w| = n$ (see Example 6, and Lemma 15).

To resolve this last difference, we extend the 2GSM with the possibility to rewrite the symbols on its input tape. In this way it can record some of the choices made. Of course, in this way we obtain the machine model of the linear space bounded Turing machine. Clearly that model is far too strong for our needs: it can accept nonregular languages in its input (whereas both 2NGSM and NMSOS have regular domains). To match the finitary nature of the nondeterministic MSO definable transduction, we restrict the machine by allowing only a bounded number of visits to each of the tape positions.

It should be clear how to extend our basic two-way machine model, and hence the 2GSM, to allow for writing on the input tape. The instructions of the extended 2GSM are of the form $(p, \sigma/\tau,\ q, \alpha, \epsilon)$ where the components are interpreted as for the 2GSM, except for σ/τ which indicates that σ on the input tape is rewritten into τ.

A computation of a two-way machine is called *k-visiting* if each of the positions of the input tape is visited at most k times. The two-way machine $\mathcal{M}$ is called *finite-visit*, if there is a constant k such that, for each pair (w, z) in the transduction realized by $\mathcal{M}$, there exists a k-visiting computation for (w, z).

Note that our definition is rather weak, as the machine may have many computations that are not k-visiting, either without any chance of reaching the final state, or with loops in the computation that produce no output.

By our choices, introducing global and bounded nondeterminism for the 2GSM, we obtain the *Hennie machine*, introduced in Hennie [1965] as an accepting device, and considered as transducer in [Rajlich 1975] (under the name 'bounded crossing transducer'). By definition, a Hennie machine is a rewriting 2GSM that is finite-visit. We find it, somewhat disguised, in Greibach [1978b] as "one way finite visit preset Turing machine," where the "preset working tape" should be interpreted as input tape, and the "one way input tape" as output tape. It is shown in Hennie [1965] that the Hennie machine accepts regular languages only.

The classes of string transductions realized by nondeterministic and deterministic Hennie machines are denoted by NHM and DHM, respectively.

Example 8. Once again consider our running nondeterministic example (cf. Example 6)

$$\{(a^n, w\#w) \mid n \geq 0, w \in \{a, b\}^*, |w| = n\}.$$

It can be realized by a Hennie machine moving in two consecutive left-to-right passes over the input. First it nondeterministically rewrites the input a^n into a string w with $|w| = n$, while writing this string to the output tape; then it writes w again to the output, copying it from the rewritten input tape. Obviously, the machine is 3-visit.

If a deterministic 2GSM visits a position of the input tape twice in the same state, then the computation will enter an infinite loop. This implies the well-known fact that every deterministic 2GSM is finite-visit, where we choose for k the number of states of the machine: 2DGSM $\subseteq$ DHM.

7.1 Visiting Sequences

It is well known (e.g., see Fischer [1969], Aho and Ullman [1970], Chytil and Jákl [1977] , Greibach [1978b;1978c], and Gurari [1982]) that the computation of a Hennie machine on an input tape can be coded as a string of "visiting sequences" (strongly related to "crossing sequences," cf. [Rabin 1963; Hennie 1965; Hopcroft and Ullman 1979; Birget 1996]). This string records, for each separate cell of the input tape, the consecutive visits of the machine to that position. We recall how this can be done, without going into too many details.

Example 9. Consider again our running example

$$\{(a^{i_1}ba^{i_2}b\cdots a^{i_n}ba^{i_{n+1}}, a^{i_1}b^{i_1}a^{i_2}b^{i_2}\cdots a^{i_n}b^{i_n}a^{i_{n+1}}) \mid n \geq 0, i_1, \ldots, i_{n+1} \geq 0\}.$$

Here we realize it with a deterministic Hennie machine, based on the three-pass method we have used in Example 1. The Hennie machine, however, rewrites on the second pass (when returning to the start of a segment of a's) each of the symbols a into b. This will allow the machine to directly move to the next nonempty segment of a's, as this simply starts at the first a to the right. The instructions are as follows.

$$\begin{array}{lll}(1, a/a,\ 1, a, +1), & (1, b/b,\ 2, \lambda, -1), & (1, \dashv/\dashv,\ 4, \lambda, 0),\\ (2, a/b,\ 2, b, -1), & (2, \sigma/\sigma,\ 3, \lambda, +1), \sigma = \vdash, b, & \\ (3, a/a,\ 1, a, +1), & (3, \sigma/\sigma,\ 3, \lambda, +1), \sigma = \vdash, b, & (3, \dashv/\dashv,\ 4, \lambda, 0)\end{array}$$

As in Example 1, the initial state is 3, and the final state is 4. Note that in state 3 the machine is looking for the next a to the right, rather than skipping a segment of a's.

We consider several types of visits during a computation, differing in the direction (-1, 0, or $+1$) of the steps taken by the machine just before and just after the visit. Additionally, a visit may be either the first or the last visit of the computation.

Given a computation of a Hennie machine, the *visiting sequence* of a position of the input tape is the sequence that starts with the symbol σ initially on the tape, followed by the consecutive visits of the machine to that position, each of them followed by the new input symbol. Each of the visits is given as a 4-tuple $({}^{-}\epsilon, p, {}^{+}\epsilon, \alpha)$ consisting of the direction ${}^{-}\epsilon$ of the move before the visit, the state p during the visit, the direction ${}^{+}\epsilon$ of the move after the visit, and the string α written to the output during that move. For the first visit of the computation we take ${}^{-}\epsilon = *$; for the last visit we take ${}^{+}\epsilon = *$ and $\alpha = \lambda$. The 4-tuple is followed by the input symbol written during the move after the visit.

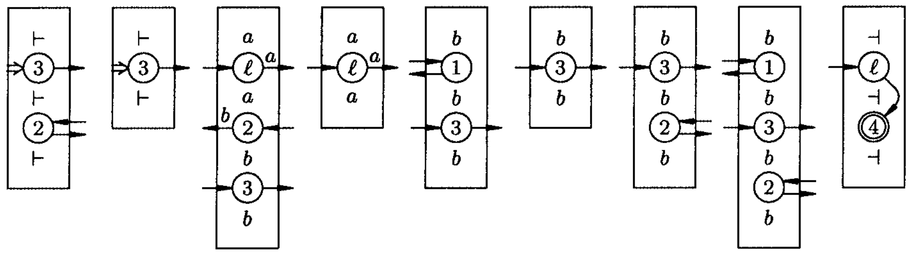


Fig. 7. Visiting sequences for Example 9, $\ell = 1, 3$.

We illustrate this notion.

Example 10. Consider the Hennie machine from Example 9. Each of the visiting sequences during a successful computation is one of the following.

$\langle \vdash,\ (*, 3, +1, \lambda),\ \vdash,\ (-1, 2, +1, \lambda),\ \vdash\ \rangle$
$\langle \vdash,\ (*, 3, +1, \lambda),\ \vdash \rangle$
$\langle\, a,\ (+1, \ell, +1, a),\ a\ , (-1, 2, -1, b),\ b\ , (+1, 3, +1, \lambda),\ b\ \rangle, \ell = 1, 3,$
$\langle\, a,\ (+1, \ell, +1, a),\ a\ \rangle, \ell = 1, 3,$
$\langle\, b,\ (+1, 1, -1, \lambda),\ b,\ (+1, 3, +1, \lambda),\ b\ \rangle$
$\langle\, b,\ (+1, 3, +1, \lambda),\ b\ \rangle$
$\langle\, b,\ (+1, 3, +1, \lambda),\ b,\ (-1, 2, +1, \lambda),\ b\ \rangle$
$\langle\, b,\ (+1, 1, -1, \lambda),\ b,\ (+1, 3, +1, \lambda),\ b,\ (-1, 2, +1, \lambda),\ b\ \rangle$
$\langle \dashv,\ (+1, \ell, 0, \lambda),\ \dashv,\ (0, 4, *, \lambda),\ \dashv \rangle, \ell = 1, 3,$

These visiting sequences are depicted in a suitable graphical manner in Figure 7.

Each visiting sequence must satisfy some syntactical constraints.

First, the directions of the visits are "alternating." This means that the first visit enters from the left (${}^{-}\epsilon = +1$, with the exception for $\sigma = \vdash$ which starts in the initial state with ${}^{-}\epsilon = *$); then, if the move after the ith visit equals ${}^{+}\epsilon = -1, 0, +1$, then the move prior to the $(i+1)$st visit to the same position must equal ${}^{-}\epsilon' = +1, 0, -1$, respectively. Only the last visit of a sequence can have ${}^{+}\epsilon = *$, in case the state is final, signalling the end of a computation.

Secondly, each visit with its surrounding input tape symbols must match an instruction of the machine. Thus, if $\langle \ldots\ \sigma,\ ({}^{-}\epsilon, p, {}^{+}\epsilon, \alpha),\ \tau,\ \ldots \rangle$ is such a visit in the visiting sequence, there must be an instruction $(p, \sigma/\tau,\ q, \alpha, {}^{+}\epsilon)$. Additionally, when ${}^{+}\epsilon = 0$, the new state q given by the instruction must match the next visit of the visiting sequence.

Clearly, also consecutive visiting sequences for a given computation must satisfy several constraints. If a visiting sequence has k "crossings" to the right, either outgoing visits $({}^{-}\epsilon, p, +1, \alpha)$ or incoming visits $(-1, p, {}^{+}\epsilon, \alpha)$—they alternate—then the visiting sequence to its right has exactly k matching crossings to the left, matching both in direction (which implicitly follows from the restrictions on single visiting sequences above) and in state change for the machine. Note that a visit $(-1, p, +1, \alpha)$ represents two crossings.

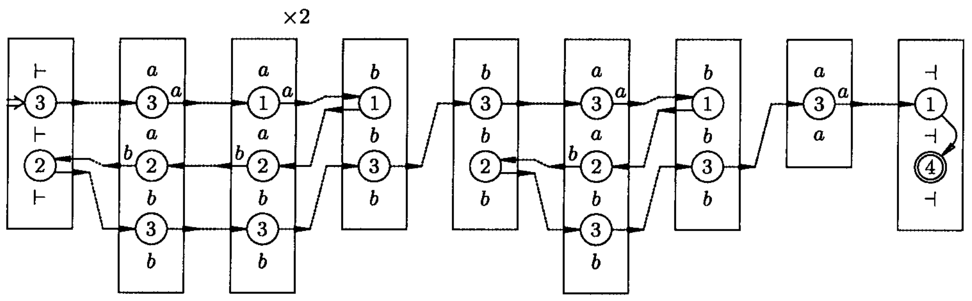


Fig. 8. Track for $\vdash a^3b^2aba\dashv$, Example 9.

Finally, the first visiting sequence of a computation should start with a visit $(*, q_{in}, {}^{+}\epsilon, \alpha)$, and exactly one visiting sequence should end with a visit $({}^{-}\epsilon, q_f, *, \lambda)$.

Since the number of visits to each position is bounded, the visiting sequences come from a finite set, and we can interprete these sequences as symbols from a finite alphabet. Each k-visiting computation is specified by a string over this alphabet, and we will call these strings *k-tracks*, e.g., the 3-track in Figure 8 specifies the computation of the Hennie machine of Example 9 on input a^3b^2aba. It should be obvious from the above remarks that the language of such specifications is regular (e.g., see Lemma 2.2 of Greibach [1978c], or Lemma 1 of Chytil and Jákl [1977]). For instance, it is the heart of the proof in Hopcroft and Ullman [1979, Theorem 2.5] of the result that two-way finite-state automata are equivalent to their one-way counterparts [Rabin and Scott 1959; Shepherdson 1959].

PROPOSITION 23. *Let $\mathcal{M}$ be a Hennie machine, and let k be a constant. The k-tracks for successful k-visiting computations of $\mathcal{M}$ form a regular language.*

7.2 Characterizations Using Hennie Machines

From Proposition 23, using standard techniques (e.g., see Chytil and Jákl [1977, Lemma 1]) we obtain the following decomposition of nondeterministic Hennie transductions. Note that this decomposition already features in Theorem 20 as characterization of NMSOS.

LEMMA 24. NHM $\subseteq$ MREL $\circ$ 2DGSM = NMSOS.

PROOF. Let $\mathcal{M}$ be a Hennie machine, finite-visit for constant k; each pair (w, z) in the transduction realized by $\mathcal{M}$ can be computed by a k-visiting computation.

We may decompose the behavior of $\mathcal{M}$ on input w as follows. First, a relabeling of $\vdash w \dashv$ guesses a string of k-visiting sequences, one for each position of the input tape, such that the first symbol of each visiting sequence matches the input symbol of the corresponding tape position. Then, a 2DGSM verifies in a left-to-right scan whether the string specifies a valid computation, a track, of $\mathcal{M}$ for w, cf. Proposition 23. If this is the case, the 2DGSM returns to the left end marker $\vdash$ and simulates $\mathcal{M}$ on this input, following the k-visiting computation previously guessed.

When changing from one tape position to a neighboring position, the 2DGSM records in its state the "crossing number" of that move, i.e., the number of times it crossed the border between these two tape positions (in one direction or another). The crossing number can be computed by inspecting the directions of the moves stored in the visiting sequence. It is used to "enter" the next visiting sequence at the correct visit, cf. the dotted lines in Figure 8. □

We now show our machine characterization of the nondeterministic MSO definable string transductions.

THEOREM 25. NMSOS = NHM.

PROOF. In view of Theorem 20 it suffices to prove the equality NHM = MREL $\circ$ 2DGSM.[3] The inclusion NHM $\subseteq$ MREL $\circ$ 2DGSM was proved in Lemma 24.

The reverse inclusion is almost immediate. In two phases the Hennie machine may simulate the composition, first writing the image of the marked relabeling on the tape, and then simulating the 2DGSM on this new tape. There is a minor technicality: for a given input w the initial tape contains $\vdash w \dashv$, and the Hennie machine is supposed to overwrite this string with its relabeling and add two new tape markers (for the simulation of the 2DGSM). Instead, it keeps the relabeling of the tape markers in its finite-state memory, rather than overwriting them. □

The demonstration of the inclusion MREL $\circ$ 2DGSM $\subseteq$ NHM in the proof of Theorem 25 can easily be extended to a proof of NHM $\circ$ NHM $\subseteq$ NHM. A Hennie machine can simulate the composition of two of its colleagues by writing the visiting sequences of the first machine onto the input tape. The output tape is contained in this string, conveniently folded over the input tape, ready to be used by the second machine.

We have, however, the closure of NHM under composition for free as a consequence of the above characterization and Proposition 17.

In Chytil and Jákl [1977] it is noted that the inclusion DHM $\circ$ 2DGSM $\subseteq$ 2DGSM can be proved analogously to their result that 2DGSM $\circ$ 2DGSM $\subseteq$ 2DGSM (i.e., 2DGSM is closed under composition, Proposition 2). That of course implies the equality of the classes of transductions realized by deterministic Hennie machines and those realized by deterministic 2GSM's. This equality is rephrased as follows.

THEOREM 26. DMSOS = DHM.

PROOF. Every 2DGSM can be considered a deterministic Hennie machine that does not rewrite its input. Hence, in view of Theorem 13 we have the inclusion DMSOS = 2DGSM $\subseteq$ DHM.

Conversely, using the characterization of Theorem 25 we have DHM $\subseteq$ NHM = NMSOS. Observe that every string transduction in DHM is a function.

[3]Restating Theorem 25 as NHM = MREL $\circ$ 2DGSM, it generalizes the result of Rajlich [1975, Theorem 2.1] that the output languages of nondeterministic Hennie machines equal the output languages of 2DGSM's; see also Greibach [1978c, Theorem 2.15(2)].

According to Theorem 21, the functions in NMSOS form exactly DMSOS. Consequently, DHM $\subseteq$ DMSOS. □

7.3 Finite-Visit Two-Way Generalized Sequential Machines

Recall that the families NMSOS and 2NGSM are incomparable (Corollary 16). However, as shown next, every nondeterministic MSO definable string transduction can be decomposed into two such transductions that can both be realized by a nondeterministic 2GSM (which is, in fact, finite-visit). Let $2\text{NGSM}_{\text{fin}}$ denote the class of string transductions realized by finite-visit 2GSM's (i.e., nondeterministic Hennie machines that do not rewrite their input). Thus, $2\text{NGSM}_{\text{fin}} \subseteq \text{NHM}$. Note also that $2\text{DGSM} \subseteq 2\text{NGSM}_{\text{fin}}$ (cf. the discussion just before Section 7.1).

THEOREM 27. $\text{NMSOS} = 2\text{NGSM}_{\text{fin}} \circ 2\text{NGSM}_{\text{fin}}$.

PROOF. By Theorem 25, $2\text{NGSM}_{\text{fin}} \subseteq \text{NHM} = \text{NMSOS}$. As the right-hand side of this inclusion is closed under composition (Proposition 17) we have the inclusion $2\text{NGSM}_{\text{fin}} \circ 2\text{NGSM}_{\text{fin}} \subseteq \text{NMSOS}$.

According to Theorem 20, NMSOS equals MREL $\circ$ 2DGSM. The inclusion from left to right follows from the fact that both $\text{MREL} \subseteq 2\text{NGSM}_{\text{fin}}$ and $2\text{DGSM} \subseteq 2\text{NGSM}_{\text{fin}}$. □

It is instructive to note that this characterization implies the (apparently new) result that $2\text{NGSM}_{\text{fin}} \circ 2\text{NGSM}_{\text{fin}}$ is closed under composition. This should be contrasted to the fact that $2\text{NGSM}_{\text{fin}}$ itself is not closed under composition. This follows from the observation from the preceding section, that the relation m from Example 6 does not belong to $2\text{NGSM} \supseteq 2\text{NGSM}_{\text{fin}}$ (Lemma 15). As we have observed, it can be realized as a combination of two 2GSM's, the first one nondeterministically changing a string a^n to a string $w \in \{a, b\}^*$ with $|w| = n$, the second one duplicating w into $w\#w$. Both of these 2GSM's are finite visit. (Alternatively, by Example 6, $m \in$ NMSOS which equals $2\text{NGSM}_{\text{fin}}^2$ as we just have seen.)

The classes 2DGSM, $2\text{NGSM}_{\text{fin}}$, and $2\text{NGSM}_{\text{fin}}^2$ form a hierarchy of transductions. However, as far as their output languages are concerned (ranges, or equivalently, with regular input) these three classes are equally powerful [Kiel 1975; Greibach 1978b].

We now show the, maybe surprising, fact that if a nondeterministic MSO definable string transduction can be realized by a 2GSM, then the 2GSM is necessarily finite-visit. This is based on the fact that the transduction is finitary, by an argument similar to the proof that every 2DGSM is finite-visit.

LEMMA 28. *Let $\mathcal{M}$ be a* 2GSM, *and let m be the transduction realized by $\mathcal{M}$. Then $\mathcal{M}$ is finite-visit iff m is* MSO *definable iff m is finitary.*

PROOF. For the implications from left to right, recall that $2\text{NGSM}_{\text{fin}} \subseteq$ NMSOS (Theorem 27) and that transductions in NMSOS are finitary because the number of parameter valuations is finite.

As for the remaining implication, assume that the finitary transduction m is realized by a 2GSM $\mathcal{M}$. As in the proof of Theorem 22, if during a (successful) computation for $(w, z) \in m$, $\mathcal{M}$ visits the same position twice in the same state, then it did not write symbols to the output in the meantime, because otherwise $\mathcal{M}$ has infinitely many output strings for the present input. Hence we may omit this loop from the computation. Consequently, there is a computation of $\mathcal{M}$ for (w, z) that does not visit each of the tape positions more than k times, where k is the number of states of $\mathcal{M}$. Hence $\mathcal{M}$ is finite-visit. □

The first part of Lemma 28 shows that $\mathrm{2NGSM}_{\mathrm{fin}}$ is exactly the class of MSO definable 2GSM transductions.

THEOREM 29. $\mathrm{2NGSM} \cap \mathrm{NMSOS} = \mathrm{2NGSM}_{\mathrm{fin}}$.

The second part of Lemma 28 shows that a 2GSM string transduction is MSO definable if and only if it is finitary. This generalizes a similar result of Courcelle [1994, Proposition 6.1] for rational transductions (i.e., string transductions realized by 2GSM's never moving to the left).

Finally we show that MSO definability is decidable for 2GSM's.

THEOREM 30. *It is decidable whether or not a given* 2GSM *transduction is* MSO *definable.*

PROOF. Let $\mathcal{M}$ be a 2GSM, realizing the transduction m. According to the proof of Lemma 28, m is MSO definable iff $\mathcal{M}$ has no successful computation in which it visits the same input position twice in the same state, writing at least one symbol to the output in the meantime. It is straightforward to construct a two-way finite-state automaton $\mathcal{M}'$ that accepts all strings obtained from some $\vdash w \dashv$ by additionally marking one position, such that, on input w, $\mathcal{M}$ has a successful computation with the above property, for that position. Since $\mathcal{M}'$ accepts a regular language [Rabin and Scott 1959; Shepherdson 1959] (see also Hopcroft and Ullman [1979, Exercise 2.18]), it now suffices to decide whether that language is empty. □

Finale. In this section we have obtained a rather precise characterization of MSO definable string transductions in terms of Hennie transductions, both in the deterministic and in the nondeterministic case. Intuitively an important reason for this equivalence is the inherent boundedness of both types of transductions: MSO definable transductions have a bound on the number of copies and the number of parameter valuations, whereas Hennie machines have a bound on the number of visits to each of the tape positions.

In case of determinism these two classes are equal to the class of transductions realized by two-way generalized sequential machines, Theorem 13. This should be contrasted to nondeterministic transductions, where 2GSM's are unable to record choices made during the computation, whereas Hennie machines may use their tape for this purpose.

We summarize.

THEOREM 31.

(1) $\mathrm{DMSOS} = \mathrm{DHM} = \mathrm{2DGSM}$.

(2) $\mathrm{NMSOS} = \mathrm{NHM} = \mathrm{MREL} \circ \mathrm{2DGSM} = \mathrm{2NGSM}^2_{\mathrm{fin}}$.

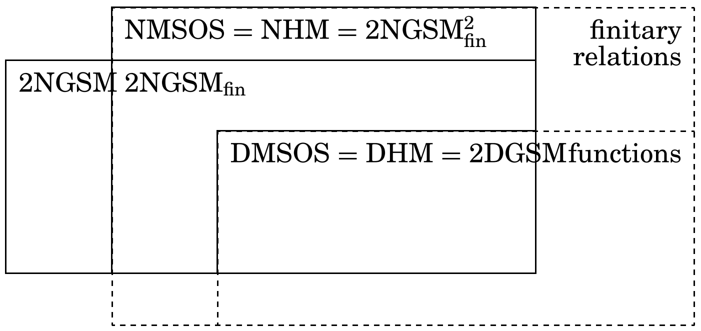

$\text{NMSOS} = \text{NHM} = \text{2NGSM}^2_{\text{fin}}$
finitary relations
2NGSM
$\text{2NGSM}_{\text{fin}}$
DMSOS = DHM = 2DGSM
functions

Fig. 9. Relationships between our main classes of transductions.

A Venn diagram of our main classes of transductions is given in Figure 9. It illustrates the results from Corollary 16, Theorems 21 and 22, Lemma 28, and Theorem 29.

Now that the classes NMSOS and 2NGSM have shown to be incomparable, unlike their deterministic counterparts, one may look for natural variants of the classes that have the same power. For machines we have discussed such a variant. Indeed, by extending the model with the power of rewriting its input tape (and at the same time demanding the finite-visit property) we obtain the Hennie transductions. We leave it as an open problem how to introduce a variant of nondeterminism for MSO definable transductions that corresponds to nondeterministic 2GSM. Additionally, we did not consider transductions realized by one-way transducers. Another remaining problem of interest is the power of first-order logic to define string transductions (where, in Definition 1, we assume all formulas to be first-order; see Example 4). Note that even for $C = \{1\}$ there are first-order definable string transductions that cannot be realized by one-way transducers (such as transforming a string into its reversal). The class of deterministic first-order definable string transductions (with respect to nd-gr) such that $C = \{1\}$ and $\phi_*^{1,1}(x, y) = \text{edge}_*(x, y)$ is characterized in Lautemann et al. [1999] to be the class of all transductions that can be realized by functional aperiodic nondeterministic one-way sequential machines (where a sequential machine is a GSM that outputs exactly one symbol at each step). The equivalence of aperiodic finite-state automata and first-order logic was established in McNaughton and Papert [1971].

ACKNOWLEDGMENTS

We thank the editor Wolfgang Thomas and the two referees for their comments which led to a major improvement of the presentation of our results. In particular, Theorem 21 and its use in the proof of Theorem 26 was suggested by one of the referees. We thank Krzysztof Apt for his "gentle nudge."

REFERENCES

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. 1969. A general theory of translation. *Math. Syst. Theor. 3*, 193–221.

AHO, A. V. AND ULLMAN, J. D. 1970. A characterization of two-way deterministic classes of languages. *J. Comput. Syst. Sci. 4*, 523–538.

BIRGET, J.-C. 1996. Two-way automata and length-preserving homomorphisms. *Math. Syst. Theor. 29*, 191–226.

BLOEM, R. AND ENGELFRIET, J. 2000. A comparison of tree transductions defined by monadic second order logic and by attribute grammars. *J. Comput. Syst. Sci. 61*, 1–50.

BÜCHI, J. R. 1960. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math. 6*, 66–92.

BÜCHI, J. R. 1962. On a decision method in restricted second order arithmetic. In *Proc. Int. Congr. Logic, Methodology and Philosophy of Sciences 1960*. Stanford University Press, Stanford, CA.

CHOMSKY, N. 1956. Three models for the description of language. *IRE Transactions on Information Theory 2*, 113–124.

CHYTIL, M. P. AND JÁKL, V. 1977. Serial composition of 2-way finite-state transducers and simple programs on strings. In *Automata, Languages and Programming, Fourth Colloquium*, A. Salomaa and M. Steinby, Eds. Lect. Notes Comput. Sci., vol. 52. Springer Verlag, Berlin, 135–147.

CLARKE, E. M., GRUMBERG, O., AND PELED, D. A. 1999. *Model Checking*. MIT Press, Cambridge, MA, USA.

COURCELLE, B. 1991. The monadic second-order logic of graphs V: On closing the gap between definability and recognizability. *Theor. Comput. Sci. 80*, 153–202.

COURCELLE, B. 1992. The monadic second-order logic of graphs VII: Graphs as relational structures. *Theor. Comput. Sci. 101,* 1, 3–33.

COURCELLE, B. 1994. Monadic second-order definable graph transductions: a survey. *Theor. Comput. Sci. 126*, 53–75.

COURCELLE, B. 1997. 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 Publishing Co., 313–400.

COURCELLE, B. AND ENGELFRIET, J. 1995. A logical characterization of the sets of hypergraphs defined by hyperedge replacement grammars. *Math. Syst. Theor. 28*, 515–552.

DONER, J. 1970. Tree acceptors and some of their applications. *J. Comput. Syst. Sci. 4*, 406–451.

EBINGER, W. 1995. Logical definability of trace languages. In *The Book of Traces*, V. Diekert and G. Rozenberg, Eds. World Scientific, 382–390. Appendix to Chapter 10.

ELGOT, C. C. 1961. Decision problems of finite automata design and related arithmetics. *Trans. AMS 98*, 21–52.

ENGELFRIET, J. 1977. Top-down tree transducers with regular look-ahead. *Math. Syst. Theor. 10*, 289–303.

ENGELFRIET, J. 1982. Three hierarchies of transducers. *Math. Syst. Theor. 15*, 95–125.

ENGELFRIET, J. 1991a. A characterization of context-free NCE graph languages by monadic second-order logic on trees. In *Graph Grammars and Their Application to Computer Science*, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, Eds. Lect. Notes Comput. Sci., vol. 532. Springer Verlag, Berlin, 311–327.

ENGELFRIET, J. 1991b. Iterated stack automata and complexity classes. *Inf. Comput. 95*, 21–75.

ENGELFRIET, J. 1997. Context-free graph grammars. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Vol. 3: Beyond Words. Springer Verlag, Berlin, 125–213.

ENGELFRIET, J. AND HEYKER, L. M. 1991. The string generating power of context-free hypergraph grammars. *J. Comput. Syst. Sci. 43*, 328–360.

ENGELFRIET, J. AND HOOGEBOOM, H. J. 1999. Two-way finite state transducers and monadic second-order logic. In *26-th International Colloquium on Automata, Languages and Programming*, J. Wiedermann, P. van Emde Boas, and M. Nielsen, Eds. Lect. Notes Comput. Sci., vol. 1644. Springer Verlag, Berlin, 311–320.

ENGELFRIET, J. AND MANETH, S. 1999. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inf. Comput. 154*, 34–91.

ENGELFRIET, J. AND VAN OOSTROM, V. 1997. Logical description of context-free graph languages. *J. Comput. Syst. Sci. 55*, 489–503.

FISCHER, M. J. 1969. Two characterizations of the context-sensitive languages. In *IEEE Conference Record of 10th Annual Symposium on Switching and Automata Theory*. 149–156.

GREIBACH, S. A. 1978a. Hierarchy theorems for two-way finite state transducers. *Acta Inf. 11*, 89–101.

GREIBACH, S. A. 1978b. One way finite visit automata. *Theor. Comput. Sci. 6*, 175–221.

Greibach, S. A. 1978c. Visits, crosses, and reversals for nondeterministic off-line machines. *Inf. and Con. 36*, 174–216.

Griffiths, T. V. 1968. The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines. *J. ACM 15*, 409–413.

Gurari, E. M. 1982. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM J. Comput. 11,* 3, 448–452.

Hennie, F. C. 1965. One-tape, off-line turing machine computations. *Inf. and Con. 8*, 553–578.

Hodges, W. 1993. *Model Theory*. Encyclopedia of Mathematics and its Applications, vol. 42. Cambridge University Press.

Hoogeboom, H. J. and ten Pas, P. 1997. Monadic second-order definable text languages. *ACM Trans. Comput. Syst. 30*, 335–354.

Hopcroft, J. E. and Ullman, J. D. 1967. An approach to a unified theory of automata. *The Bell System Technical Journal 46*, 1793–1829. Also in: IEEE Conference Record of 8th Annual Symposium on Switching and Automata Theory, Austin, Texas, 1967, pages 140–147.

Hopcroft, J. E. and Ullman, J. D. 1979. *Introduction to Automata Theory, Language, and Computation*. Addison–Wesley, Reading, Mass.

Immerman, N. 1999. *Descriptive Complexity*. Springer Verlag, Berlin.

Kiel, D. 1975. Two-way a-transducers and AFL. *J. Comput. Syst. Sci. 10*, 88–109.

Kleene, S. C. 1956. Representation of events in nerve nets and finite automata. In *Automata Studies*. Annals of Mathematics Studies, vol. 34. Princeton University Press, Princeton, N.J., 3–42.

Kurshan, R. P. 1994. *Computer-Aided Verification of Coordinated Processes*. Princeton University Press, Princeton, N.J.

Lapoire, D. 1998. Recognizability equals monadic second-order definability for sets of graphs of bounded tree-width. In *15th Annual Symposium on Theoretical Aspects of Computer Science*. Lect. Notes Comput. Sci., vol. 1373. Springer Verlag, Berlin, 618–628.

Lautemann, C., McKenzie, P., Schwentick, T., and Vollmer, H. 1999. The descriptive complexity approach to LOGCFL. In *16th Symposium on Theoretical Aspects of Computer Science*, C. Meinel and S. Tison, Eds. Lect. Notes Comput. Sci., vol. 1563. Springer Verlag, Berlin, 444–454. Full version: Electronic Colloquium on Computational Complexity, Report TR98-059.

McCulloch, W. S. and Pitts, W. 1943. A logical calculus of the ideas imminent in nervous activity. *Bull. Math. Bioph. 5*, 115–133.

McNaughton, R. and Papert, S. 1971. *Counter-free automata*. MIT Press, Cambridge, MA.

Myhill, J. 1957. Finite automata and the representation of events. Tech. Rep. WADD TR-57-624, Wright Patterson AFB, Ohio.

Nerode, A. 1958. Linear automata transformation. *Proc. AMS 9*, 541–544.

Nijholt, A. 1982. The equivalence problem for LL- and LR-regular grammars. *J. Comput. Syst. Sci. 24*, 149–161.

Pixton, D. 1996. Regularity of splicing languages. *Discr. Appl. Math. 69*, 101–124.

Rabin, M. O. 1963. Real-time computation. *Isr. J. Math 1*, 203–211.

Rabin, M. O. 1969. Decidability of second-order theories and automata on infinite trees. *Trans. AMS 141*, 1–35.

Rabin, M. O. 1977. Decidable theories. In *Handbook of Mathematical Logic*, J. Barwise, Ed. Studies in Logic and the Foundations of Mathematics, vol. 30. North-Holland, 595–629.

Rabin, M. O. and Scott, D. 1959. Finite automata and their decision problems. *IBM J. Res. Dev. 3*, 114–125. Also in: E.F. Moore, Ed., *Sequential Machines: Selected Papers*. Addison-Wesley, Reading, MA, 1964, pages 63–91.

Rajlich, V. 1975. Bounded-crossing transducers. *Inf. and Con. 27*, 329–335.

Seese, D. 1992. Interpretability and tree automata: a simple way to solve algorithmic problems on graphs closely related to trees. In *Tree Automata and Languages*, M. Nivat and A. Podelski, Eds. Elsevier Science Publishers, Amsterdam, 83–114.

Shepherdson, J. C. 1959. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev. 3*, 198–200. Also in: E.F. Moore, Ed., *Sequential Machines: Selected Papers*. Addison-Wesley, Reading, MA, 1964, pages 92–97.

Thatcher, J. W. and Wright, J. B. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Syst. Theor. 2*, 57–82.

Thomas, W. 1997a. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science—A Selection of Essays in Honor of A. Ehrenfeucht*, J. Mycielski, G. Rozenberg, and A. Salomaa, Eds. Lect. Notes Comput. Sci., vol. 1261. Springer Verlag, Berlin, 118–143.

Thomas, W. 1997b. Languages, automata, and logic. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Vol. 3: Beyond Words. Springer Verlag, Berlin, 389–455.

Trakhtenbrot, B. A. 1962. Finite automata and the logic of one-place predicates. *Siberian Mathematical Journal 3*, 103–131 (in Russian). English translation: *American Mathematical Society Translations, Series 2*, 59 (1966), 23–55.

Yu, S. 1997. Regular languages. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Vol. 1: Word, Language, Grammar. Springer Verlag, Berlin, 41–110.

Received June 1999; revised September 2000; accepted October 2000