

Bruno Courcelle

---

Graph Structure and Monadic  
Second-Order Logic

July 2009

to be published by

Cambridge University Press



# Contents

<b>8</b>	<b>Monadic second-order transductions of terms and words</b>	<b>1</b>
8.1	Terminology . . . . .	3
8.2	More on automata and logic . . . . .	5
8.2.1	Automata . . . . .	5
8.2.2	Logic . . . . .	6
8.2.3	Logic and automata . . . . .	8
8.3	Tree-walking tree transducers . . . . .	8
8.4	The basic characterization . . . . .	13
8.5	From jumping to walking . . . . .	15
8.6	From global to local tests . . . . .	17
8.7	Nondeterminism . . . . .	24
8.8	VR-equational sets of terms and words . . . . .	26
8.9	Bibliographic remarks . . . . .	29
8.9.1	Words . . . . .	29
8.9.2	Terms . . . . .	30



## Chapter 8

# Monadic second-order transductions of terms and words

As explained in the introduction of Section 1.7, there do not exist finite-state automata or finite-state transducers that work directly on graphs. Thus, for graphs, the role of automata and transducers is taken over by monadic second-order logic: instead of accepting a set of graphs by an automaton, it is defined by an MS-sentence; and instead of computing a graph transformation by a transducer, it is defined as an MS-transduction by a definition scheme. With respect to automata, the original motivation for this approach was Theorem 1.16 (and its converse, cf. Theorem 5.66): for terms and words MS logic and finite-state automata have the same expressive power.

The aim of this chapter is to show that, in a certain sense, the automata-theoretic characterization of monadic-second order logic for sets of terms and words (Theorem 1.16 and its converse) can be generalized to transductions. This means of course, that the automata should produce output, i.e., that they are transducers. We will concentrate on transductions that are partial functions, and in particular on deterministic devices, i.e., parameterless definition schemes and deterministic transducers. For these, we will show that MS-transductions of words correspond to two-way finite-state transducers, and that MS-transductions of terms correspond to (compositions of) tree-walking tree transducers. These transducers are well known from Formal Language Theory.

A two-way finite-state transducer is a finite-state automaton with a two-way read-only input tape and a one-way output tape. The input word is placed on the input tape between endmarkers, and the transducer has a reading head that scans one cell of the input tape at a time. In one move, it reads the symbol in that cell, moves one cell to the left or the right (or stays in the same cell), writes some symbols on the output tape, and goes into another state. Such a

transducer is said to be a one-way finite-state transducer if it never moves to the left.

A tree-walking tree transducer is very similar to a two-way finite-state transducer, but more complicated as both its input and output are trees (or more precisely, syntactic trees of terms). It is a finite-state device, of which the reading head scans a node of the input tree, at each moment of time; this node will be called the “current node”. In one move, it reads the label of that node and either does not produce output, or produces one labelled node of the output tree. In the first case, it moves to the father or to one of the sons of the current node (or stays at the same node), and goes into another state. In the second case, the automaton transforms into  $k$  parallel copies of itself, where  $k$  is the number of sons of the produced output node. Each of these copies then moves to the father or one of the sons of the current node (or stays where it is), and goes into another state. Different copies can move to different input nodes and go into different states; in other words, the copies behave independently (and will behave independently in the future). The task of the  $i$ -th copy is to output the tree rooted at the  $i$ -th son of the produced output node. Thus, the output tree is generated in a top-down fashion, with several copies of the transducer at some of the leaves of the output generated so far. The computation of such a copy ends when an output node is produced that has no sons. It should be noted that a tree-walking transducer can detect whether or not the current node has a father (i.e., whether it is the root), which son it is of its father, and how many sons it has itself. We also note, for readers familiar with alternation, that the “copying” behaviour of a nondeterministic tree-walking tree transducer is similar to that of an alternating automaton.

We will identify words with monadic trees, i.e., trees of which each node has at most one son, and the (unique) leaf has a specific label  $\epsilon$ . A word  $a_1 a_2 \cdots a_n$  is identified with (the syntactic tree of) the term  $a_1(a_2(\cdots a_n(\epsilon)\cdots))$ . A tree-walking transducer that translates monadic trees into monadic trees is, apart from notational matters, the same as a two-way finite-state transducer. The symbol  $\epsilon$  acts as a right endmarker; there is no left endmarker, but the transducer can detect the left end of the word because it is the root of the monadic tree. Walking up and down on the monadic input tree corresponds to the two-way (left and right) motion on the input tape. Since the output tree is monadic, the transducer never transforms into two or more independent copies, and therefore the output is produced as on a one-way output tape.

One might expect or hope that MS-transductions of words correspond to the more usual one-way finite-state transducers that define the so-called rational transductions (see, e.g., [Ber79]). However, these two classes of transductions have different properties. As observed in Section 1.7, the class of rational transductions is closed under composition and inverse, and rational transductions preserve regularity and context-freeness of languages. On the other hand, the class of MS-transductions of words is closed under composition (by Theorem 7.7) but not under inverse. Moreover, inverse MS-transductions of words preserve regularity of languages but MS-transductions do not. MS-transductions do not preserve context-freeness either, but (by Corollary 7.33) they do preserve

VR-equationality (of which context-freeness is a, proper, special case). These statements will be made more precise in what follows.

Through the Recognizability Theorem (Section 5.3.8), finite-state automata on terms can be viewed as an implementation of the specification of sets of graphs by MS-sentences. For describing terms, the implementation formalism fits the specification formalism, in the sense that they have the same expressive power. Arbitrary MS-transductions are complicated to understand and to implement. This chapter shows how to implement MS-transductions of terms as finite-state transducers. In the case of words, the two-way finite-state transducer fits the MS-transduction, in the above sense (see Theorem 8.12). For arbitrary terms, MS-transductions can be implemented by the composition of two tree-walking tree transducers, the first of which never moves up (see Theorem 8.18). This implementation formalism fits the MS-transduction when it is appropriately restricted (see Theorem 8.19). For terms we also discuss other implementation formalisms: the tree-walking pushdown tree transducer (see Theorem 8.15) and the macro tree transducer (see Theorem 8.27). After the automata-theoretic characterization of MS-transductions of terms and words, we obtain from the Equationality Theorem (Theorem 7.32) automata-theoretic characterizations of the VR-equational (equivalently, HR-equational) sets of terms and words (see Theorems 8.25 and 8.26).

## 8.1 Terminology

For the reader's convenience, we review some definitions and notation, and introduce some new terminology, to be used in this chapter.

For binary relations  $R_1$  and  $R_2$ , their composition is  $R_2 \circ R_1 = \{(x, z) \mid \exists y : (x, y) \in R_1, (y, z) \in R_2\}$ . For classes  $X_1$  and  $X_2$  of binary relations,  $X_2 \circ X_1 = \{R_2 \circ R_1 \mid R_1 \in X_1, R_2 \in X_2\}$ .

In this chapter, all *alphabets* (for words) are assumed to be *finite*. Moreover, all *functional signatures* are assumed to be *finite and one-sorted*, and all *relational signatures* are assumed to be *without constants*.

For a functional signature  $F$  we denote by  $\rho(f)$  the arity of  $f \in F$ , and by  $\rho(F)$  the maximal arity of an element of  $F$ . For  $k \in \mathbb{N}$ ,  $F_k = \{f \in F \mid \rho(f) = k\}$ . The set of all (well-formed) terms over  $F$  is denoted  $T(F)$ . A term will also be called a *tree*; that should not lead to confusion, because in this chapter we will not consider trees other than syntactic trees of terms.

For a term  $t \in T(F)$ ,  $root_t$  denotes its root, and for every node (or position)  $u$  of  $t$ ,  $u_i$  denotes the  $i$ -th son of  $u$  (if it has one). The root has depth 1, and the depth of  $u_i$  is one more than the depth of  $u$ . Furthermore,  $t/u$  denotes the subtree at node  $u$  and  $t \uparrow u$  denotes the context at  $u$  (with  $w$  as context variable).

With a functional signature  $F$  we associate the relational signature  $\mathcal{R}_F = \{lab_f \mid f \in F\} \cup \{son_i \mid i \in [\rho(F)]\} \cup \{rt\}$  where  $\rho(lab_f) = 1$ ,  $\rho(son_i) = 2$ , and  $\rho(rt) = 1$ . A term  $t \in T(F)$  is represented by the structure

$$[t] = \langle N_t, (lab_{ft})_{f \in F}, (son_{it})_{i \in [\rho(F)]}, rt_t \rangle \in STR(\mathcal{R}_F)$$

where  $N_t$  is the set of nodes of  $t$ ,  $lab_{ft} = Occ(t, f)$ ,  $son_{it}$  consists of all  $(u, v) \in N_t^2$  such that  $v$  is the  $i$ -th son of  $u$ , and  $rt_t = \{root_t\}$ . Without  $rt$ , this definition of  $\mathcal{R}_F$  and  $\lfloor t \rfloor$  is the same as in the beginning of Section 6.3.3. It is equivalent with the one given in Section 5.1.1. Note that we have taken the  $rt$  relation to be of arity 1, to avoid relational signatures with constants. In fact (as observed in Section 5.1.1), the relation  $rt$  is superfluous, because it is FO-expressible in terms of the others:  $rt(x) \Leftrightarrow \neg \exists y. \bigvee_{i \in [\rho(F)]} son_i(y, x)$ . Thus, whenever we define a monadic second-order transduction to  $T(F)$ , we need not (and usually will not) specify the formulas  $\theta_{(rt, (i))}$  that define the  $rt$  relation in the output structure.

In this chapter *we view words as a special case of terms*, viz. monadic terms. In this way we avoid the duplication of definitions and proofs. A functional signature is *monadic* if it is unary (i.e., all its elements have arity 0 or 1), and it has exactly one element of arity 0, denoted  $\epsilon$ . For an alphabet  $A$ , there is a natural bijection  $\mu_A : A^* \rightarrow T(U_A)$ , where  $U_A$  is the monadic functional signature  $A \cup \{\epsilon\}$  ( $\epsilon \notin A$ ) with  $\rho(a) = 1$  for every  $a \in A$  and  $\rho(\epsilon) = 0$ : the word  $a_1 a_2 \cdots a_n$  corresponds to the term  $\mu_A(a_1 a_2 \cdots a_n) = a_1(a_2(\cdots a_n(\epsilon)\cdots))$ . In fact,  $\mu_A$  is the inverse of the mapping  $val_{\mathbb{W}_{left}(A)}$ , see Definition 2.7. Obviously, as stated in Remark 3.20(4), a language  $L \subseteq A^*$  is regular if and only if the set of monadic terms  $\mu_A(L) \subseteq T(U_A)$  is regular.

A word  $w \in A^*$  is represented by the structure  $\lfloor w \rfloor := \lfloor \mu_A(w) \rfloor$  in  $STR(\mathcal{R}_{U_A})$ . In this chapter it is important that the empty word  $\varepsilon$  is represented by a structure with a singleton as domain rather than by the empty structure. The reason is that an MS-transduction transforms the empty structure into itself, whereas it can transform a singleton structure into any other structure. For MS-transductions, the definition of  $\lfloor w \rfloor$  in Section 5.1.1 is equivalent to the above definition, provided  $\lfloor \varepsilon \rfloor$  is defined to have a singleton as domain.

We will consider MS-transductions of terms and words. By Propositions 5.22 and 5.23 these are the same as the CMS-transductions of terms and words. But we will only consider deterministic devices, i.e., parameterless definition schemes and deterministic transducers; to stress this, we will also use ‘deterministic’ instead of ‘parameterless’. Note that parameterless MS-transductions are partial functions. We denote by DMSOW the class of parameterless MS-transductions of words (where the ‘D’ stands for ‘deterministic’ whence ‘parameterless’), by DMSOT the class of parameterless MS-transductions of terms, and by DMSOTW those from terms to words. Thus, DMSOT is the class of all transductions  $\tau : T(F) \rightarrow T(H)$  such that  $\{(\lfloor [s] \rfloor_{iso}, \lfloor [t] \rfloor_{iso}) \mid (s, t) \in \tau\}$  is a parameterless MS-transduction from  $STR(\mathcal{R}_F)$  to  $STR(\mathcal{R}_H)$ , and similarly for the other two classes; in the remainder of this chapter we will be less precise and write  $(\lfloor s \rfloor, \lfloor t \rfloor)$  instead of  $(\lfloor [s] \rfloor_{iso}, \lfloor [t] \rfloor_{iso})$ . It follows from (the proof of) Theorem 7.7 that DMSOW and DMSOT are closed under composition.

An arbitrary transduction  $\tau$  (of structures) is of *linear size increase* if the size of the output is linear in the size of the input (where ‘size’ refers to the cardinality of the domains), i.e., if there is an integer  $k$  such that  $|D_T| \leq k \cdot |D_S|$  for every  $(S, T) \in \tau$ . Obviously, as observed just before Example 7.2 (and

in Fact 1.36), every MS-transduction is of linear size increase, where  $k$  is the copying number of the definition scheme by which it is defined. A transduction  $\tau$  has finite images if  $\tau(S) = \{T \mid (S, T) \in \tau\}$  is finite (up to isomorphism) for every input structure  $S$ . Since, for given  $n$ , there are only finitely many structures  $T$  such that  $|D_T| \leq n$  (up to isomorphism), every transduction of linear size increase has finite images; in particular every MS-transduction has finite images.

To define the semantics of a tree-walking tree transducer we will use regular grammars (Definition 3.17). A *regular (tree) grammar* is a triple  $G = (F, X, R)$  where  $F$  is a functional signature,  $X$  is a finite set of nullary symbols, called nonterminals, disjoint with  $F$ , and  $R$  is a set of rules of the form  $x \rightarrow t$  with  $x \in X$  and  $t \in T(F, X)$ . A regular grammar generates terms over  $F$ , i.e.,  $L(G, x) \subseteq T(F)$  for every  $x \in X$ . In fact, a regular grammar is a particular context-free grammar. The one-step derivation relation of  $G$  is denoted  $\Rightarrow_G$ , hence  $L(G, x) = \{t \in T(F) \mid x \Rightarrow_G^* t\}$ .

## 8.2 More on automata and logic

We need some more terminology on tree automata, and some more facts on monadic second-order logic and regular sets of terms.

### 8.2.1 Automata

Recall from Section 3.3.1 that, given a functional signature  $F$ , a (bottom-up,  $\varepsilon$ -free) *finite automaton* over  $F$  (or *finite  $F$ -automaton*) is a tuple  $\mathcal{A} = \langle F, Q, \delta, Acc \rangle$  where  $Q$  is a finite set of states,  $Acc \subseteq Q$  is the set of accepting states, and  $\delta$  is a set of transition rules of the form  $f[q_1, \dots, q_k] \rightarrow q$  with  $f \in F_k$  and  $q, q_1, \dots, q_k \in Q$  (if  $k = 0$ , the transition rule is also written  $f \rightarrow q$ ). Recall that  $\mathcal{A}$  is *complete and deterministic* if for every  $f \in F_k$  and  $q_1, \dots, q_k \in Q$  there is exactly one transition rule  $f[q_1, \dots, q_k] \rightarrow q$  in  $\delta$  (with  $q \in Q$ ). This definition is based on viewing  $\mathcal{A}$  as a bottom-up device.

In Section 8.6 we will also view  $\mathcal{A}$  as a (usual) *top-down* tree automaton. In that case, intuitively,  $Acc$  is the set of initial states (because  $\mathcal{A}$  starts at the root of the input tree), and a transition rule  $f[q_1, \dots, q_k] \rightarrow q$  in  $\delta$  is interpreted from right to left: if  $\mathcal{A}$  is in state  $q$  at node  $u$  with label  $f$ , then  $\mathcal{A}$  transforms into  $k$  copies of itself, and the  $i$ -th copy moves to the  $i$ -th son of  $u$  in state  $q_i$ . With this view in mind, we say that  $\mathcal{A}$  is *top-down complete and deterministic* if  $Acc$  is a singleton and for every  $q \in Q$  and  $f \in F_k$  with  $k \neq 0$  there is exactly one transition rule  $f[q_1, \dots, q_k] \rightarrow q$  in  $\delta$  (with  $q_1, \dots, q_k \in Q$ ).

Let  $\mathcal{A} = \langle F, Q, \delta, Acc \rangle$  be a complete and deterministic finite automaton; for  $f \in F_k$  and  $q_1, \dots, q_k \in Q$  we denote by  $\delta_f(q_1, \dots, q_k)$  the unique state  $q$  such that  $f[q_1, \dots, q_k] \rightarrow q$  is a transition in  $\delta$ . For  $s \in T(F)$  and  $u \in Pos(s)$  we define  $state_{s, \mathcal{A}}(u) \in Q$ , the state in which  $\mathcal{A}$  reaches  $u$ , recursively as follows: if  $u \in Occ(s, f)$  with  $\rho(f) = k$ , then  $state_{s, \mathcal{A}}(u) = \delta_f(state_{s, \mathcal{A}}(u_1), \dots, state_{s, \mathcal{A}}(u_k))$ . Obviously  $state_{s, \mathcal{A}}(u) = run_{\mathcal{A}, s}(u)$ , where  $run_{\mathcal{A}, s}$  is the unique run of  $\mathcal{A}$  on  $s$ .

(cf. Lemma 3.56). Thus, the set of terms accepted by  $\mathcal{A}$  is  $L(\mathcal{A}) = \{s \in T(F) \mid \text{state}_{s,\mathcal{A}}(\text{root}_s) \in \text{Acc}\}$ .

Now let  $\mathcal{A} = \langle F, Q, \delta, \text{Acc} \rangle$  be a top-down complete and deterministic finite automaton. For  $s \in T(F)$  and  $u \in \text{Pos}(s)$  we define  $\text{tstate}_{s,\mathcal{A}}(u) \in Q$ , the state in which  $\mathcal{A}$  reaches  $u$  top-down, recursively as follows:  $\text{tstate}_{s,\mathcal{A}}(\text{root}_s)$  is the state  $q$  such that  $\text{Acc} = \{q\}$ , and if  $u \in \text{Occ}(s, f)$  with  $\rho(f) = k \neq 0$ , then, for every  $i \in [k]$ ,  $\text{tstate}_{s,\mathcal{A}}(u_i)$  is the state  $q_i$  such that  $f[q_1, \dots, q_k] \rightarrow q$  is the unique rule in  $\delta$  with  $q = \text{tstate}_{s,\mathcal{A}}(u)$ .

A set of terms is *regular* if it can be generated by a regular grammar (from some nonterminal), or, alternatively, if it can be accepted by a (complete deterministic) finite automaton, cf. Propositions 3.48 and 3.49. The class of regular sets of terms is denoted REGT.

## 8.2.2 Logic

Let  $F$  be a functional signature. To implement an MS-transduction of terms by a tree-walking transducer, the transducer will need to know, at each node  $u$  of a tree  $s \in T(F)$  which MS formulas  $\varphi(x_1)$  are true at  $u$ , and in some cases even which formulas  $\varphi(x_1, x_2)$  are true at  $u$  and some other node. To handle this formally, we define the following theories, similar to, but different from those in the proof of the Equationality Theorem (Theorem 7.24).

Let  $q \in \mathbb{N}$  be given; it will be specified whenever we need the definitions that follow. For  $n \in \mathbb{N}$ , let  $L_n := \widehat{MS^q}(\mathcal{R}_F, \{x_1, \dots, x_n\})$  and  $L'_n := \widehat{MS^q}(\mathcal{R}_{F \cup \{w\}}, \{x_1, \dots, x_n\})$ , where  $w$  is the unique variable used for contexts. For  $s \in T(F)$  and  $u \in N_s$ , we define

$$\begin{aligned} Th_s^\downarrow(u) &= \{\varphi \in L_1 \mid [s/u] \models \varphi(u)\}, \\ Th_s^\uparrow(u) &= \{\varphi \in L'_1 \mid [s \uparrow u] \models \varphi(u)\}, \text{ and} \\ Th_s(u) &= \{\varphi \in L_1 \mid [s] \models \varphi(u)\}. \end{aligned}$$

and for  $s \in T(F)$  and  $a, u \in N_s$ , we define

$$\begin{aligned} Th_s^\downarrow(a, u) &= \{\varphi \in L_2 \mid [s/u] \models \varphi(a, u)\} \quad \text{provided } a \in N_{s/u}, \\ Th_s^\uparrow(a, u) &= \{\varphi \in L'_2 \mid [s \uparrow u] \models \varphi(a, u)\} \quad \text{provided } a \in N_{s \uparrow u}, \text{ and} \\ Th_s(a, u) &= \{\varphi \in L_2 \mid [s] \models \varphi(a, u)\}. \end{aligned}$$

Note that if  $m \leq n$  then  $MS^q(\mathcal{R}_F, \{x_1, \dots, x_m\}) \subseteq MS^q(\mathcal{R}_F, \{x_1, \dots, x_n\})$ , and similarly for  $\mathcal{R}_{F \cup \{w\}}$ . Hence, e.g.,  $Th_s^\downarrow(u)$  contains all sentences  $\hat{\varphi}$  such that  $\varphi \in MS^q(\mathcal{R}_F, \emptyset)$  and  $[s/u] \models \varphi$ .

We now use the Splitting Theorem for derived operations (Theorem 5.47), and in particular Application 5.51, to obtain a number of relationships between these theories. It is easy to see that the results of Application 5.51 are also valid with our definition of  $\mathcal{R}_F$  and  $[t]$  (with  $rt$  of arity 1 instead of 0).

Let  $u$  be a node of  $s \in T(F)$  with label  $f \in F_k$  and with sons  $u_1, \dots, u_k$ . Then (see Application 5.51), there is a mapping  $\delta_f : \mathcal{P}(L_1)^k \rightarrow \mathcal{P}(L_1)$  (independent from  $s$  and  $u$ ), such that

$$Th_s^\downarrow(u) = \delta_f(Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k)).$$

We will express this by saying that  $Th_s^\downarrow(u)$  can be determined from  $f$  and  $Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k)$ . Similarly (see again Application 5.51), there is a mapping  $Z_f : \mathcal{P}(L'_1) \times \mathcal{P}(L_1)^k \rightarrow \mathcal{P}(L_1)$ , such that

$$Th_s(u) = Z_f(Th_s^\uparrow(u), Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k)),$$

i.e.,  $Th_s(u)$  can be determined from  $f$ ,  $Th_s^\uparrow(u)$ , and  $Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k)$ . Moreover, by a similar argument, for every  $i \in [k]$ , there is a mapping  $Z_{f,i} : \mathcal{P}(L'_1) \times \mathcal{P}(L_1)^{k-1} \rightarrow \mathcal{P}(L'_1)$ , such that

$$Th_s^\uparrow(u_i) = Z_{f,i}(Th_s^\uparrow(u), Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_{i-1}), Th_s^\downarrow(u_{i+1}), \dots, Th_s^\downarrow(u_k)),$$

in other words,  $Th_s^\uparrow(u_i)$  can be determined from  $f$ ,  $Th_s^\uparrow(u)$ , and all  $Th_s^\downarrow(u_j)$  with  $j \in [k]$ ,  $j \neq i$ .

Similar statements are valid when two nodes  $a$  and  $u$  are considered. For instance, if  $a \in N_{s \uparrow u}$ , then  $Th_s(a, u)$  can be determined from  $f$ ,  $Th_s^\uparrow(a, u)$ , and  $Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k)$ . As another example, if  $a \in N_{s/u_i}$ , then  $Th_s^\downarrow(a, u)$  can be determined from  $f$ ,  $Th_s^\downarrow(a, u_i)$ , and all  $Th_s^\downarrow(u_j)$  with  $j \neq i$ ; moreover, for every  $j \neq i$ ,  $Th_s^\uparrow(u_j)$  can be determined from  $f$ ,  $Th_s^\uparrow(u)$ ,  $Th_s^\downarrow(a, u_i)$ , and all  $Th_s^\downarrow(u_m)$  with  $m \neq i, j$ .

We note here that the mappings  $\delta_f$  considered above can be viewed as the transition rules of a complete and deterministic finite automaton  $\mathcal{A} = \langle F, Q, \delta, Acc \rangle$  with  $Q = \mathcal{P}(L_1)$  and  $Acc$  is irrelevant. Obviously, for every  $s \in T(F)$  and  $u \in N_s$ ,  $state_{s, \mathcal{A}}(u) = Th_s^\downarrow(u)$ . Hence, if  $\varphi \in L_0$  has quantifier-height  $\leq q$  and  $Acc$  consists of all subsets of  $L_1$  that contain  $\hat{\varphi}$ , then  $L(\mathcal{A}) = \{s \in T(F) \mid s \models \varphi\}$  (see Corollary 5.56).

We finally recall from the proof of the Equationality Theorem (Theorem 7.24), that for every  $\Phi \subseteq L_1$  there exists an MS formula  $\eta_\Phi^\downarrow(x_1)$  such that  $\lfloor s \rfloor \models \eta_\Phi^\downarrow(u)$  if and only if  $Th_s^\downarrow(u) = \Phi$ , and a similar formula  $\eta_\Phi^\uparrow(x_1)$  exists for  $Th_s^\uparrow(u) = \Phi$ . The formula  $\eta_\Phi^\downarrow(x_1)$  is

$$\exists X [\forall y (y \in X \Leftrightarrow y \leq x_1) \wedge \bigwedge_{\varphi \in \Phi} \varphi'(X, x_1) \wedge \bigwedge_{\varphi \in L_1 - \Phi} \neg \varphi'(X, x_1)]$$

where  $y \leq x_1$  expresses that  $y$  is a descendant of  $x_1$  (see Proposition 5.9) and  $\varphi'(X, x_1)$  is obtained from  $\varphi(x_1)$  by the relativization of  $\varphi(x_1)$  to  $X$  and the replacement of every atomic formula  $rt(x)$  by  $x = x_1$ . Similarly, the formula  $\eta_\Phi^\uparrow(x_1)$  is

$$\exists X [\forall y (y \in X \Leftrightarrow \neg y \leq x_1) \wedge \bigwedge_{\varphi \in \Phi} \varphi'(X, x_1) \wedge \bigwedge_{\varphi \in L'_1 - \Phi} \neg \varphi'(X, x_1)]$$

where  $\varphi'(X, x_1)$  is obtained from  $\varphi(x_1)$  by the relativization of  $\varphi(x_1)$  to  $X$  and the replacement of every atomic formula  $lab_w(x)$  by  $x = x_1$  and every atomic formula  $lab_f(x)$ , with  $f \neq w$ , by  $lab_f(x) \wedge x \neq x_1$ .

### 8.2.3 Logic and automata

We recall the basic connection between monadic second-order logic and automata, for sets of words and terms. It was stated in Theorem 1.16 (for terms) and proved in Theorem 5.13 and Corollary 5.56 (and stated again in Theorem 5.66). Taking the terms to be monadic, the result for words is obtained.

**Theorem 8.1** Let  $F$  be a functional signature.

A set of terms  $L \subseteq T(F)$  is regular if and only if there exists a sentence  $\varphi$  in  $\text{MS}(\mathcal{R}_F, \emptyset)$  such that  $L = \{t \in T(F) \mid \llbracket t \rrbracket \models \varphi\}$ .

One consequence of this result is that inverse MS-transductions preserve regularity (as observed for words in the introduction to this chapter). In fact, using Theorem 1, the following two statements are immediate from Corollary 7.6 and Proposition 7.17, respectively.

**Proposition 8.2** Let  $\tau : T(F) \rightarrow T(H)$  be an MS-transduction of terms.

1. If  $L \subseteq T(H)$  is regular, then  $\tau^{-1}(L) \subseteq T(F)$  is regular.
2. If  $\tau$  is an MS-relabelling and  $L \subseteq T(F)$  is regular, then  $\tau(L) \subseteq T(H)$  is regular.

## 8.3 Tree-walking tree transducers

To facilitate the proof of the characterization of MS-transductions of terms through tree-walking tree transducers, we consider a hybrid device: a tree-walking tree transducer that has MS tests and MS jumps, i.e., that uses monadic second-order logic for two purposes: for a given input term  $s$ , (1) it can use a formula  $\varphi(x_1)$  to test a global property of the current node  $x_1$  of  $s$ , i.e., a property of the pair  $(s, x_1)$  rather than just a (local) property of  $x_1$  (such as its label), and (2) it can use a formula  $\psi(x_1, x_2)$  to describe conditions implying a jump in  $s$  from the current node  $x_1$  to the new current node  $x_2$ , that need not be the father or a son of  $x_1$  (or  $x_1$  itself). This hybrid transducer allows us to prove the characterization in several steps (rather than in one long proof). To stress the role of determinism, we also consider nondeterministic tree-walking tree transducers, and prove some of the results on transducers also for the nondeterministic case.

An *MS tree-walking tree transducer* (abbreviated *mstwt*) is a tuple  $M = (F, H, Q, q_{\text{in}}, R)$ , where  $F$  and  $H$  are functional signatures of input and output symbols, respectively,  $Q$  is a finite set of states,  $q_{\text{in}} \in Q$  is the initial state, and  $R$  is a finite set of rules. A rule is of the form  $\langle q, \varphi \rangle \rightarrow \zeta$  with  $q \in Q$ ,  $\varphi \in \text{MS}(\mathcal{R}_F, \{x_1\})$ , and  $\zeta$  is of one of the following two forms:

1.  $\langle q', \psi \rangle$ ,
2.  $h(\langle q_1, \psi_1 \rangle, \dots, \langle q_k, \psi_k \rangle)$ ,

where (1)  $q' \in Q$  and  $\psi \in \text{MS}(\mathcal{R}_F, \{x_1, x_2\})$  such that  $[s] \models \forall x_1[\varphi(x_1) \Rightarrow \exists! x_2. \psi(x_1, x_2)]$  for every  $s \in T(F)$ , or (2)  $h \in H_k$ ,  $q_1, \dots, q_k \in Q$ , and  $\psi_1, \dots, \psi_k \in \text{MS}(\mathcal{R}_F, \{x_1, x_2\})$  such that  $[s] \models \forall x_1[\varphi(x_1) \Rightarrow \exists! x_2. \psi_i(x_1, x_2)]$  for every  $s \in T(F)$  and every  $i \in [k]$ . As usual the quantifier  $\exists!$  stands for “there exists exactly one”. Rules of type 1 will be called *epsilon-output* rules.

Note that the above conditions on  $\varphi, \psi, \psi_1, \dots, \psi_k$  are decidable by Theorem 8.1 and the decidability of the emptiness of a regular set of terms (which is a context-free language). Thus, our syntax for mstwts is effective.

The mstw  $M$  is *deterministic* if for every two distinct rules  $\langle q_1, \varphi_1 \rangle \rightarrow \zeta_1$  and  $\langle q_2, \varphi_2 \rangle \rightarrow \zeta_2$  in  $R$ , if  $q_1 = q_2$  then  $\varphi_1$  and  $\varphi_2$  are mutually exclusive on  $T(F)$ , i.e.,  $[s] \models \neg \exists x_1[\varphi_1(x_1) \wedge \varphi_2(x_1)]$  for every  $s \in T(F)$ . Again, note that this is a decidable condition.

For every input term  $s \in T(F)$  we define a regular grammar  $G_{M,s} = (H, X_{M,s}, R_{M,s})$  over the output signature  $H$ . The set  $X_{M,s}$  of nonterminals consists of all pairs  $\langle q, u \rangle$  with  $q \in Q$  and  $u \in N_s$  (i.e.,  $u$  is a node of  $s$ ). Such a pair is called a *configuration* of  $M$  on  $s$ . If  $\langle q, \varphi \rangle \rightarrow \zeta$  is a rule in  $R$ , then  $R_{M,s}$  contains a rule  $\langle q, u \rangle \rightarrow \zeta'$  for every  $u \in N_s$  such that  $[s] \models \varphi(u)$ , where  $\zeta'$  equals

1.  $\langle q', v \rangle$ , and  $v$  is the unique node such that  $[s] \models \psi(u, v)$ ,
2.  $h(\langle q_1, v_1 \rangle, \dots, \langle q_k, v_k \rangle)$ , and  $v_i$  is the unique node such that  $[s] \models \psi_i(u, v_i)$  for every  $i \in [k]$ ,

respectively (according to the two possible forms of  $\zeta$  mentioned above). The computations of  $M$  on input  $s$  are the derivation sequences of the grammar  $G_{M,s}$ . The *transduction computed by  $M$* , denoted  $\tau_M$ , is defined as  $\tau_M = \{(s, t) \in T(F) \times T(H) \mid t \in L(G_{M,s}, \langle q_{\text{in}}, \text{root}_s \rangle)\}$ .

Note that if  $M$  is deterministic then distinct rules of  $G_{M,s}$  have distinct left-hand sides. This implies that  $L(G_{M,s}, \langle q_{\text{in}}, \text{root}_s \rangle)$  is either empty or a singleton, and hence  $\tau_M$  is a (partial) function.

An mstw  $M$  is *single-use* if for every  $s \in T(F)$ ,  $\xi \in T(H, X_{M,s})$ ,  $t \in T(H)$ , and  $\langle q, u \rangle \in X_{M,s}$ , if  $\langle q_0, \text{root}_s \rangle \Rightarrow_{G_{M,s}}^* \xi \Rightarrow_{G_{M,s}}^* t$  then  $\langle q, u \rangle$  occurs at most once in  $\xi$ . Note that every mstw with a monadic output signature is single-use. The class of transductions computed by single-use deterministic MS tree-walking tree transducers is denoted  $\text{DTWT}_{\text{su}}^{\text{MS}}$ .

The mstw  $M$  is called an *MS tree-walking tree-to-word transducer* if  $H$  is monadic. If  $B = H - \{\epsilon\}$ , then  $M$  realizes the following tree-to-word transduction:  $\{(s, w) \in T(F) \times B^* \mid (s, \mu_B(w)) \in \tau_M\}$ . In this case one may view  $M$  as having a one-way output tape, on which the output word is written. The class of transductions computed by deterministic MS tree-walking tree-to-word transducers is denoted  $\text{DTWTW}^{\text{MS}}$ .

The mstw  $M$  is called an *MS two-way finite-state transducer* if  $F$  and  $H$  are monadic. If  $A = F - \{\epsilon\}$  and  $B = H - \{\epsilon\}$ , then  $M$  realizes the following word transduction:  $\{(v, w) \in A^* \times B^* \mid (\mu_A(v), \mu_B(w)) \in \tau_M\}$ .

The mstw  $M$  is an *MS tree-walking automaton* if  $H = \{\epsilon\}$ , with  $\rho(\epsilon) = 0$ . Then the *tree language recognized by  $M$*  is  $L(M) = \{s \in T(F) \mid (s, \epsilon) \in \tau_M\}$ .

An mstwt is “jumping” rather than “walking”; moreover, it can do “global” rather than “local” tests (cf. the introduction of this section). We say that the mstwt  $M$  is *nonjumping* if the only formulas  $\psi(x_1, x_2)$  that are used in the right-hand sides of rules are  $x_1 = x_2$  (stay where you are),  $son(x_2, x_1)$  (move to the father), and  $son_i(x_1, x_2)$  (move to the  $i$ -th son) for every  $i \in [\rho(F)]$ . These formulas will also be denoted stay, up, and down $_i$ , respectively. Note that  $son(x_2, x_1)$  abbreviates  $\bigvee_{i \in [\rho(F)]} son_i(x_2, x_1)$ . We say that  $M$  is *local* if it is nonjumping and the only formulas  $\varphi$  that are used in the left-hand sides of rules are boolean combinations of  $rt(x_1)$ ,  $lab_f(x_1)$ , and  $\exists y. son_i(y, x_1)$ , for every  $f \in F$  and  $i \in [\rho(F)]$ ; note that  $rt(x_1)$  is, in fact, superfluous. In our terminology, locality will be indicated by dropping ‘MS’. Thus, a local mstwt is also called a *tree-walking tree transducer* (for short *twt*), and the class of transductions computed by deterministic twts is denoted DTWT (and by DTWT<sub>su</sub> for single-use deterministic twts). The class of transductions computed by deterministic tree-walking tree-to-word transducers is denoted DTWTW, and the class of (word) transductions computed by deterministic two-way finite-state transducers is denoted 2DGSM (where ‘2gsm’ stands for ‘two-way generalized sequential machine’, a well-known name for two-way finite-state transducers).

In the remainder of this section we consider examples of MS-transductions of words and terms that can be computed by tree-walking tree transducers.

**Example 8.3** The *yield* of a term  $t \in T(F)$  is the (nonempty) word  $yd(t) \in F_0^*$  defined recursively as follows:  $yd(f) = f$  for  $f \in F_0$ , and  $yd(f(t_1, \dots, t_k)) = yd(t_1) \cdots yd(t_k)$  for  $f \in F_k$  with  $k > 0$  and  $t_1, \dots, t_k \in T(F)$ . The mapping  $yd$  is a 2-copying parameterless MS-transduction in DMSOTW: the positions of the output word  $yd(t)$  are the leaves of the input tree  $t$ , plus one extra position (e.g., the second copy of the last leaf). Each (first copy of a) leaf is connected to (the first copy of) the next leaf, in the left-to-right order of the nodes of  $t$ , and the first copy of the last leaf is connected to its second copy (which gets label  $\epsilon$ ).

The mapping  $yd$  can also be computed by a (local) deterministic tree-walking tree-to-word transducer  $M = (F, H, Q, q_{in}, R)$ . The transducer walks depth-first left-to-right through the input tree and outputs the labels of the leaves; it outputs  $\epsilon$  when it returns to the root. Thus,  $yd$  is in DTWTW. To consider a concrete example, let  $A$  be a finite set. We take  $F = \{f\} \cup A$  with  $\rho(f) = 2$  and  $\rho(a) = 0$  for  $a \in A$ , and we take  $H = A \cup \{\epsilon\}$  with  $\rho(a) = 1$  for  $a \in A$  and  $\rho(\epsilon) = 0$ . The set of states of  $M$  is  $Q = \{\downarrow, \uparrow_1, \uparrow_2\}$  and  $q_{in} = \downarrow$ . The rules in  $R$  are the following, for  $a \in A$  and  $i \in \{1, 2\}$  (and  $br_i(x_1)$  abbreviates  $\exists y. son_i(y, x_1)$ ):

$$\begin{array}{ll}
\langle \downarrow, lab_f(x_1) \rangle & \rightarrow \langle \downarrow, down_1 \rangle \\
\langle \downarrow, lab_a(x_1) \wedge br_i(x_1) \rangle & \rightarrow a(\langle \uparrow_i, up \rangle) \\
\langle \uparrow_1, True \rangle & \rightarrow \langle \downarrow, down_2 \rangle \\
\langle \uparrow_2, br_i(x_1) \rangle & \rightarrow \langle \uparrow_i, up \rangle \\
\langle \uparrow_2, rt(x_1) \rangle & \rightarrow \epsilon \\
\langle \downarrow, lab_a(x_1) \wedge rt(x_1) \rangle & \rightarrow a(\langle \uparrow_2, stay \rangle).
\end{array}$$

Let us also consider a slightly more complicated example: let  $H' = A \cup \{\epsilon\}$  with

$\rho(a) = 2$  for  $a \in A$  and  $\rho(\epsilon) = 0$ , and define  $\tau : T(F) \rightarrow T(H')$  as follows. For every tree  $t \in T(F)$ ,  $\tau(t)$  is a full binary tree such that the sequence of labels of the nodes of any path from the root to a leaf in  $\tau(t)$  equals  $yd(t)\epsilon$ . Then  $\tau$  is computed by the deterministic tree-walking tree transducer that is obtained from  $M$  by changing the right-hand side of the second rule into  $a(\langle \uparrow_i, \text{up} \rangle, \langle \uparrow_i, \text{up} \rangle)$ , and similarly the right-hand side of the last rule into  $a(\langle \uparrow_2, \text{stay} \rangle, \langle \uparrow_2, \text{stay} \rangle)$ .

**Example 8.4** Consider the mapping  $\tau$  on words that associates with  $w \in \{a, b, c, d\}^*$  the word  $\tau(w) = a^k b^l c^m d^n$  where  $k = |w|_a$  (the number of occurrences of  $a$  in  $w$ ),  $l = |w|_b$ ,  $m = |w|_c$ , and  $n = |w|_d$ . It is not difficult to see that  $\tau$  is a noncopying parameterless MS-transduction, i.e., that there is a noncopying parameterless definition scheme  $\mathcal{D} = \langle \chi, \delta, (\theta_R)_{R \in \mathcal{R}_{FA}} \rangle$  such that  $\hat{\mathcal{D}}(\lfloor \mu_A(w) \rfloor) = \lfloor \mu_A(\tau(w)) \rfloor$  for every  $w \in A^*$ , where  $A = \{a, b, c, d\}$ . In fact, both  $\chi$  and  $\delta$  are *True*, and for every  $f \in U_A = A \cup \{\epsilon\}$ ,  $\theta_{lab_f} = lab_f(x_1)$  (which means that every node keeps its label). The details of the formula  $\theta_{son_1}(x_1, x_2)$  are left to the reader: it should connect each occurrence of  $a$  in  $w$  to the next occurrence of  $a$ , unless it is the last occurrence of  $a$ , in which case it should be connected to the first occurrence of  $b$ ; and similarly for  $b$  and  $c$ , for  $c$  and  $d$ , and for  $d$  and  $\epsilon$ .

It should also be clear that  $\tau$  can be computed by a deterministic two-way finite-state transducer. The transducer scans the input word four times (for instance from left to right and from right to left, and then again). At the first scan it outputs all occurrences of  $a$ , and at the next scans all occurrences of  $b$ ,  $c$ , and  $d$ ; finally it outputs  $\epsilon$ .

Thus,  $\tau$  is in both DMSOW and 2DGSM. Let  $L$  be the regular language  $(abcd)^*$ . Obviously,  $\tau(L) = \{a^n b^n c^n d^n \mid n \geq 0\}$ . Hence,  $\tau$  does not preserve regularity or context-freeness of languages (see the discussion in the introduction of this chapter). We know from Proposition 8.2 that  $\tau^{-1}$  preserves regularity. Now consider the context-free language  $L' = \{a^k b^k c^m d^m \mid k, m \geq 0\}$ . Then  $\tau^{-1}(L')$  is not context-free; in fact, if it would be context-free then its intersection with the regular language  $R = (ac)^* b^* d^*$  would also be context-free, but it is easy to see that  $\tau^{-1}(L') \cap R = \{(ac)^n b^n d^n \mid n \geq 0\}$ , a classical noncontext-free language. Thus, inverse MS-transductions of words do not preserve context-freeness of languages.

**Example 8.5** A transduction of words  $\tau : A^* \rightarrow B^*$  is a *rational transduction* if it can be expressed as follows:  $\tau(w) = h'(h^{-1}(w) \cap L)$  for every  $w \in A^*$ , where  $L \subseteq C^*$  is a regular language for some alphabet  $C$ , and  $h : C^* \rightarrow A^*$  and  $h' : C^* \rightarrow B^*$  are homomorphisms (for equivalent characterizations see the book by Sakarovitch [Saka03]). In other words,  $\tau = h' \circ \text{id}_L \circ h^{-1}$ , where  $\text{id}_L$  is the identity on  $L$ . It is well known (and the proof is a straightforward exercise) that all rational transductions can be computed by two-way, even one-way, finite-state transducers. In fact, the rational transductions are exactly those computed by one-way finite-state transducers (see [Saka03, Chapter 4]).

Obviously, by Theorem 8.1, the partial function  $\text{id}_L$  is a noncopying parameterless MS-transduction (take  $\chi$  to be the MS formula defining the regular

language  $L$ ). It is not difficult to show that all homomorphisms are parameterless MS-transductions (the definition scheme for  $h'$  is  $k$ -copying, where  $k$  is the maximal length of a word  $h'(c)$  with  $c \in C$ ). However, inverse homomorphisms are not MS-transductions, because, in general, they do not have finite images (whereas the MS-transductions do, see Section 8.1); for instance, if  $h(c) = \varepsilon$  for some  $c \in C$ , then  $h^{-1}(\varepsilon)$  contains all words  $c^n$ ,  $n \geq 0$ . Thus, the class of MS-transductions of words is not closed under inverse, cf. the discussion in the introduction of this chapter. If the homomorphism  $h$  is nonerasing (i.e.,  $h(c) \neq \varepsilon$  for all  $c \in C$ ), then one can rather easily show that  $h^{-1}$  is a non-copying MS-transduction: one uses a parameter  $X = \{u_1, \dots, u_n\} \subseteq Pos(w)$  where  $u_i$  is the first position of a subword  $h(c_i)$  of the input word  $w$ , such that  $w = h(c_1) \cdots h(c_n)$  for  $c_1, \dots, c_n \in C$ ; additionally, one uses parameters  $X_c$ ,  $c \in C$ , that form the partition of  $X$  such that  $X_c = \{u_i \in X \mid c_i = c\}$ ; in other words, the parameters represent a guess of a word  $v = c_1 \cdots c_n \in C^*$  such that  $h(v) = w$ ; the output word  $v$  is constructed from the positions in  $X$ , and  $u \in X$  is given the label  $c$  if  $u \in X_c$ . Thus, in this case the rational transduction  $\tau$  is an MS-transduction, by Theorem 7.7. It can be shown that every rational transduction  $\tau$  that has finite images, can be expressed as above with  $h$  nonerasing (see the book by Berstel [Ber79, Exercise 7.2, page 87], or the one by Autebert and Boasson [AutBoa88, Chapter 1]). Consequently, a rational transduction is an MS-transduction if and only if it has finite images.

**Example 8.6** Let  $\tilde{w}$  be the mirror image of the word  $w$ . The mapping  $\tau(w) = w\tilde{w}$  is in 2DGSM: the transducer first walks from left to right over the input word, and then back from right to left; it outputs each letter that it meets on its way. If it does *not* produce output during the walk from left to right, it computes the mapping  $\tau'(w) = \tilde{w}$ . Both  $\tau$  and  $\tau'$  are deterministic MS-transductions (2-copying and noncopying, respectively). Note that neither  $\tau$  nor  $\tau'$  is a rational transduction;  $\tau'$  preserves regularity and context-freeness, but  $\tau$  does not. For every  $k \geq 2$ , the mapping  $\tau_k(w) = w^k$  is in 2DGSM: the transducer walks  $k$  times from left to right over the input word, writing it to the output tape (each time walking back from right to left without producing output). Moreover,  $\tau_k$  is a  $k$ -copying parameterless MS-transduction (in DMSOW); it can obviously be expressed as  $\mu \circ copy_k$ , where  $\mu$  is a noncopying parameterless MS-transduction, cf. Proposition 7.21.

**Example 8.7** Consider the second-order substitutions, already defined in Section 2.6.1 (just before Proposition 2.125). They are usually called *tree homomorphisms*. A second-order substitution is *linear* if it corresponds to linear derived operations. It is easy to see that every second-order substitution can be computed by a (local) deterministic twt that does not use ‘up’ in the right-hand sides of its rules; for instance, if the derived operation  $h$  of arity 3 is defined by the term  $g(x_3, g'(x_1))$ , then the transducer has rules  $\langle q_{in}, lab_h(x_1) \rangle \rightarrow g(\langle q_{in}, down_3 \rangle, \langle q, stay \rangle)$  and  $\langle q, True \rangle \rightarrow g'(\langle q_{in}, down_1 \rangle)$ , where  $q$  is a “new” state. Moreover, if the second-order substitution is linear (as in the above instance), then the twt is single-use. Thus, every linear second-order substitution

is in  $\text{DTWT}_{\text{su}}$ . It is also a parameterless MS-transduction in DMSOT. Just as the word homomorphisms in Example 8.5, it has a  $k$ -copying definition scheme, where  $k$  is the maximal size of the defining terms of the involved derived operations. The details are left to the reader.

The hypothesis that tree homomorphisms are linear is important. Let  $H = \{g, a\}$  and  $F = \{f, a\}$  with  $\rho(g) = 1$ ,  $\rho(f) = 2$ , and  $\rho(a) = 0$ . Consider the (nonlinear) second-order substitution  $\theta : T(H) \rightarrow T(F)$  such that  $\theta(a) = a$  and  $\theta(g(t)) = f(\theta(t), \theta(t))$ ; in other words,  $a$  has defining term  $t_a = a$ , and  $g$  has defining term  $t_g = f(x_1, x_1)$ . Since, for every  $n \geq 1$ ,  $|\theta(g^{n-1}(a))| = 2^n - 1$ , the mapping  $\theta$  is not of linear size increase, and hence is not an MS-transduction. This shows that tree-walking tree transducers have more expressive power than MS-transductions of terms.

## 8.4 The basic characterization

We start by showing the basic characterization of MS-transductions (of terms) by transducers: they correspond to MS tree-walking tree transducers that are single-use. Intuitively, the single-use restriction means that the tree-walking transducer can at most output  $k$  copies of each input node, where  $k$  is the number of its states: due to determinism, if the transducer visits an input node twice in the same state, then it repeats itself and hence does not halt (and produces no output). This  $k$  corresponds to the copying number of a definition scheme.

**Theorem 8.8**  $\text{DMSOT} = \text{DTWT}_{\text{su}}^{\text{MS}}$ .

**Proof:** We first prove the inclusion  $\text{DMSOT} \subseteq \text{DTWT}_{\text{su}}^{\text{MS}}$ . Let  $\tau : T(F) \rightarrow T(H)$  be a deterministic MS-transduction of terms, i.e.,  $\{([s], [t]) \mid (s, t) \in \tau\}$  is a parameterless MS-transduction from  $\text{STR}(\mathcal{R}_F)$  to  $\text{STR}(\mathcal{R}_H)$ . Let  $\mathcal{D} = \langle \chi, \delta_1, \dots, \delta_k, (\theta_w)_{w \in \mathcal{R}_H \boxtimes k} \rangle$  be a parameterless definition scheme of type  $\mathcal{R}_F \rightarrow \mathcal{R}_H$  defining  $\tau$ , i.e.,  $\hat{\mathcal{D}}([s]) = [\tau(s)]$  for every  $s \in T(F)$ . We define an mstw  $M = (F, H, Q, q_{\text{in}}, R)$  that computes  $\tau$ , as follows. First, the set of states is  $Q = \{q_{\text{in}}, q_1, \dots, q_k\}$ . Intuitively, for  $i \in [k]$ , the state  $q_i$  represents the  $i$ -th copy of the input structure. More precisely, if  $M$  is in configuration  $\langle q_i, u \rangle$  on the input tree  $[s]$ , then it will output the node  $(u, i)$  of the output tree  $[\tau(s)]$ . It remains to define the set of rules  $R$  of  $M$ . We first define the rules for  $q_{\text{in}}$ . Intuitively,  $M$  starts its computation by jumping to the root of the output tree. For every  $i \in [k]$ ,  $M$  has the rule:

$$\langle q_{\text{in}}, \chi \wedge \exists x[\delta_i(x) \wedge \theta_{(rt, (i))}(x)] \rangle \rightarrow \langle q_i, \delta_i(x_2) \wedge \theta_{(rt, (i))}(x_2) \rangle.$$

Now suppose that  $M$  is in configuration  $\langle q_i, u \rangle$ . Then it outputs node  $(u, i)$  of the output tree and jumps to its sons. To do this, it has to determine the label of that node, and its sons. For every  $h \in H_m$  and every  $j_1, \dots, j_m \in [k]$ ,  $M$  has

the rule  $\langle q_i, \varphi \rangle \rightarrow h(\langle q_{j_1}, \psi_1 \rangle, \dots, \langle q_{j_m}, \psi_m \rangle)$  where  $\varphi$  is

$$\chi \wedge \delta_i(x_1) \wedge \theta_{(lab_h, (i))}(x_1) \wedge \bigwedge_{n \in [m]} \exists y [\delta_{j_n}(y) \wedge \theta_{(son_n, (i, j_n))}(x_1, y)]$$

and  $\psi_n$  is  $\delta_{j_n}(x_2) \wedge \theta_{(son_n, (i, j_n))}(x_1, x_2)$ .

It should be clear that  $M$  computes  $\tau$ , and that  $M$  is deterministic. Moreover  $M$  is single-use, because the configurations in its computation (except the first) are in bijection with the nodes of the output tree.

For the inclusion  $\text{DTWT}_{\text{su}}^{\text{MS}} \subseteq \text{DMSOT}$ , let  $M = (F, H, Q, q_{\text{in}}, R)$  be a single-use deterministic mstwt. We first consider the special case where  $M$  has no epsilon-output rules. Without loss of generality, we assume that  $Q = [k]$  for some  $k \geq 1$ .

Since  $M$  is deterministic and single-use, each node of the output tree (for a given input tree  $s$ ) is produced by a unique configuration  $\langle q, u \rangle$  of  $M$ , with  $q \in [k]$  and  $u \in N_s$ . Intuitively, this means that a  $k$ -copying definition scheme for  $\tau_M$  can be constructed that uses  $(u, q)$  for that node. To simplify the construction we will actually show that  $\tau_M$  is the composition  $\tau_3 \circ \tau_2 \circ \tau_1$  of three parameterless MS-transductions, of which the first is  $k$ -copying and the other two are noncopying. Since the class of parameterless MS-transductions is closed under composition by (the proof of) Theorem 7.7, this gives the desired result.

The first MS-transduction  $\tau_1$  translates the input tree  $s$  into the “computation space” of  $M$  on  $s$ , which is a representation by a structure  $S$  in  $\text{STR}(\mathcal{R}_H)$  of the regular grammar  $G_{M,s} = (H, X_{M,s}, R_{M,s})$  and the start configuration  $\langle q_{\text{in}}, \text{root}_s \rangle$ . The structure  $S$  has the domain  $\{(u, q) \mid \langle q, u \rangle \in X_{M,s}\}$ , and it has the unary relation  $rt_S = \{(\text{root}_s, q_{\text{in}})\}$ . Let  $\langle q, u \rangle \rightarrow h(\langle q_1, v_1 \rangle, \dots, \langle q_m, v_m \rangle)$  be a rule in  $R_{M,s}$ . Then  $(u, q)$  is in  $lab_{h,S}$ , and  $((u, q), (v_i, q_i))$  is in  $son_{i,S}$ , for every  $i \in [m]$ . Note that  $S$  represents a labelled directed graph with a designated node (the “root”).

This MS-transduction  $\tau_1$  is defined by the  $k$ -copying parameterless definition scheme  $\mathcal{D}_1 = \langle \chi, \delta_1, \dots, \delta_k, (\theta_w)_{w \in \mathcal{R}_H \boxtimes k} \rangle$  of type  $\mathcal{R}_F \rightarrow \mathcal{R}_H$  where  $\chi$  expresses the fact that the input structure represents a term in  $T(F)$ , see Corollary 5.10, and for every  $q, q' \in [k]$ ,  $\delta_q = \text{True}$ ,  $\theta_{(rt, (q))}$  is  $rt(x_1)$  if  $q = q_{\text{in}}$  and *False* otherwise,  $\theta_{(lab_h, (q))}$  is the disjunction of all formulas  $\varphi$  such that  $\langle q, \varphi \rangle \rightarrow h(\langle q_1, \psi_1 \rangle, \dots, \langle q_m, \psi_m \rangle)$  is a rule of  $M$ , and  $\theta_{(son_i, (q, q'))}$  is the disjunction of all formulas  $\varphi \wedge \psi_i$  such that  $\langle q, \varphi \rangle \rightarrow h(\langle q_1, \psi_1 \rangle, \dots, \langle q_m, \psi_m \rangle)$  is a rule of  $M$  with  $i \in [m]$  and  $q_i = q'$ .

The second MS-transduction  $\tau_2$  transforms the computation space into the subgraph induced by all vertices that are reachable from the “root” of the graph. Thus,  $\tau_2$  is defined by the noncopying parameterless definition scheme  $\mathcal{D} = \langle \chi, \delta, (\theta_R)_{R \in \mathcal{R}_H} \rangle$  of type  $\mathcal{R}_H \rightarrow \mathcal{R}_H$  with  $\chi = \text{True}$ ,  $\delta = \exists y (rt(y) \wedge \text{TC}[son; y, x_1])$ ,  $\theta_{lab_h} = lab_h(x_1)$ ,  $\theta_{son_i} = son_i(x_1, x_2)$ , and  $\theta_{rt} = rt(x_1)$ . Note that  $son = \lambda u, v. son(u, v)$ , where  $son(u, v)$  is the formula  $\bigvee_{i \in [\rho(H)]} son_i(u, v)$ ; for TC see Section 5.2.2.

The third MS-transduction  $\tau_3$  checks that the resulting graph is a tree (i.e., that  $s$  is in the domain of  $\tau_M$ ). Since  $M$  is single-use, it suffices to check that

(1) there is no circuit (otherwise  $M$ 's computation on  $s$  does not halt), and  
(2) every vertex has a label in  $H$  (otherwise  $M$ 's computation on  $s$  aborts).  
Thus,  $\tau_3$  is defined by the noncopying parameterless definition scheme  $\mathcal{D} = \langle \chi, \delta, (\theta_R)_{R \in \mathcal{R}_H} \rangle$  of type  $\mathcal{R}_H \rightarrow \mathcal{R}_H$  where  $\chi$  expresses the above properties (1) and (2), i.e., it is  $\forall x, y (\text{TC}[\text{son}; x, y] \Rightarrow \neg \text{son}(y, x)) \wedge \forall x. \bigvee_{h \in H} \text{lab}_h(x)$ , and  $\delta = \text{True}$ ,  $\theta_{\text{lab}_h} = \text{lab}_h(x_1)$ ,  $\theta_{\text{son}_i} = \text{son}_i(x_1, x_2)$ , and  $\theta_{rt} = \text{rt}(x_1)$ .

Finally, let us consider a single-use deterministic mstwt  $M = (F, H, Q, q_{\text{in}}, R)$  with epsilon-output rules. Then, let the mstwt  $M' = (F, H \cup \{\odot\}, Q, q_{\text{in}}, R')$  be obtained from  $M$  by changing every epsilon-output rule  $\langle q, \varphi \rangle \rightarrow \langle q', \psi \rangle$  of  $M$  into the output producing rule  $\langle q, \varphi \rangle \rightarrow \odot(\langle q', \psi \rangle)$  of  $M'$ , where  $\odot$  is a new output symbol of arity 1. Obviously  $M'$  is still a single-use deterministic mstwt, and thus we know that  $\tau_{M'}$  is in DMSOT. It is also obvious that  $\tau_M = \tau_{\odot} \circ \tau_{M'}$ , where  $\tau_{\odot} : T(H \cup \{\odot\}) \rightarrow T(H)$  erases the  $\odot$ -labelled nodes of the input tree. Since DMSOT is closed under composition, it now suffices to show that  $\tau_{\odot}$  is in DMSOT. That is easy: it is defined by the noncopying parameterless definition scheme  $\mathcal{D} = \langle \chi, \delta, (\theta_R)_{R \in \mathcal{R}_H} \rangle$  of type  $\mathcal{R}_H \rightarrow \mathcal{R}_H$  such that  $\chi = \text{True}$ ,  $\delta = \neg \text{lab}_{\odot}(x_1)$ ,  $\theta_{\text{lab}_h} = \text{lab}_h(x_1)$ , and  $\theta_{\text{son}_i} = \exists y (\text{son}_i(x_1, y) \wedge \text{TC}[\lambda u, v. \alpha; y, x_2])$ , where  $\alpha(u, v)$  is  $\text{lab}_{\odot}(u) \wedge \text{son}_1(u, v)$ . ■

If the output signature is monadic, then the subscript ‘su’ can be dropped from Theorem 8.8, because every mstwt with a monadic output signature is single-use. Thus:  $\text{DMSOTW} = \text{DTWTW}^{\text{MS}}$ .

We observe that the construction of a parameterless definition scheme for an mstwt  $M$ , as described in the above proof, can also be carried out in the case where  $M$  is not single-use. Then, instead of a tree, the output structure is a directed graph  $G$  without circuit (i.e., a tree with shared subtrees). Unfolding  $G$  produces the output tree of  $M$ . This also shows that it is decidable whether or not  $M$  is single-use:  $M$  is single-use if and only if the indegree of every vertex of  $G$  is at most 1, for every output  $G$ . Since this is an MS-expressible property, the set of all input terms such that  $G$  does not satisfy this property is regular by Corollary 7.6 and Theorem 8.1, and can be checked for emptiness.

## 8.5 From jumping to walking

Our aim will now be to “get rid of” the MS tests and MS jumps of our hybrid tree-walking transducer. It turns out that MS jumps can be simulated by tree-walking tree transducers, provided they are still allowed the use of MS tests. However, MS tests cannot be handled by tree-walking tree transducers; thus, for the simulation of tests other means have to be found. We therefore start with the “removal” of jumps.

**Theorem 8.9** For every mstwt an equivalent nonjumping mstwt can be constructed. Determinism and the single-use restriction are preserved.

**Proof:** Let  $M$  be an mstwt with input signature  $F$ . The problem is how to simulate a jump of  $M$ , i.e., a formula  $\psi \in \text{MS}(\mathcal{R}_F, \{x_1, x_2\})$  in the right-hand

side of a rule of  $M$ , by an ordinary walk. It is easy to see that we may assume that  $[s] \models \forall x_1 \exists! x_2. \psi(x_1, x_2)$  for every  $s \in T(F)$ . In fact, if this is not the case, then we replace  $\psi$  by  $(\varphi \wedge \psi) \vee (\neg\varphi \wedge x_1 = x_2)$ , where  $\varphi \in \text{MS}(\mathcal{R}_F, \{x_1\})$  is the formula in the left-hand side of the rule.

We will prove that for every such formula  $\psi$  a nonjumping deterministic mstwt  $M_\psi = (F, \emptyset, Q, q_{\text{in}}, R)$  can be constructed, with epsilon-output rules only, and a state  $q_f \in Q$ , such that for every  $s \in T(F)$  and  $a, b \in N_s$ ,  $[s] \models \psi(a, b)$  if and only if  $\langle q_{\text{in}}, a \rangle \Rightarrow_G^* \langle q_f, b \rangle$ , where  $G = G_{M_\psi, s}$ . It is easy to see that, using these mstwts  $M_\psi$  as subroutines,  $M$  can be turned into an equivalent nonjumping mstwt. Since every  $M_\psi$  is deterministic, determinism and the single-use restriction are preserved. Note that, since  $M_\psi$  has epsilon-output rules only, it is essentially a tree-walking automaton rather than a tree-walking transducer.

So, let  $\psi \in \text{MS}(\mathcal{R}_F, \{x_1, x_2\})$  with  $[s] \models \forall x_1 \exists! x_2. \psi(x_1, x_2)$  for every  $s \in T(F)$ . Let  $q = \text{Max}\{qh(\psi), m\} + 2$  where  $m$  is the quantifier-height of the formula  $x_1 \leq x_2$  expressing that  $x_1$  is a descendant of  $x_2$  (see Proposition 5.9). For this  $q$  (and  $F$ ), we will use the definitions in Section 8.2.2.

The mstwt  $M_\psi = (F, \emptyset, Q, q_{\text{in}}, R)$  should walk from node  $a$  to node  $b$  if and only if  $[s] \models \psi(a, b)$ . It will do this in a special way: it walks along the shortest (undirected) path in  $s$  from  $a$  to  $b$ , and, roughly speaking, at each node  $u$  on this path it keeps track in its state of the finite set of formulas  $Th_s^\downarrow(a, u)$  or  $Th_s^\uparrow(a, u)$  depending on whether or not  $a$  is a descendant of  $u$ .

More precisely,  $M_\psi$  first walks from  $a$  up to the least common ancestor  $\text{lca}(a, b)$  of  $a$  and  $b$ , and at each node  $u$  on that path (except  $a$ ) it is in state  $\uparrow_{i, \Phi}$  such that  $a$  is a descendant of the  $i$ -th son  $u_i$  of  $u$  and  $\Phi = Th_s^\downarrow(a, u_i)$ . Then,  $M_\psi$  walks from  $\text{lca}(a, b)$  down to  $b$ , and at each node  $u$  on that path (except  $\text{lca}(a, b)$ ) it is in state  $\downarrow_\Phi$  such that  $\Phi = Th_s^\uparrow(a, u)$ .

Thus, the set  $Q$  of states of  $M_\psi$  consists of  $q_{\text{in}}$ ,  $q_f$ ,  $\uparrow_{i, \Phi}$ , and  $\downarrow_\Phi$ , for all  $i \in [\rho(F)]$  and  $\Phi \subseteq L_2 \cup L_2'$ . The right-hand sides of the rules in  $R$  are of one of the three forms  $\langle \uparrow_{i, \Phi}, \text{up} \rangle$ ,  $\langle \downarrow_\Phi, \text{down}_j \rangle$ , or  $\langle q_f, \text{stay} \rangle$ . Recall that  $\text{up}$ ,  $\text{down}_j$ , and  $\text{stay}$  denote the formulas  $\text{son}(x_2, x_1)$ ,  $\text{son}_j(x_1, x_2)$ , and  $x_1 = x_2$ , respectively.

For every node  $u$  that  $M_\psi$  visits, it uses a test of the form  $\eta_\Phi^\uparrow(x_1) \wedge \eta_{\Phi_1}^\downarrow(x_1) \wedge \dots \wedge \eta_{\Phi_k}^\downarrow(x_1)$  to compute the sets of formulas  $Th_s^\uparrow(u)$  and  $Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k)$ , where  $u_1, \dots, u_k$  are the sons of  $u$  (see the end of Section 8.2.2). It should now be clear from Section 8.2.2 that from these sets, and the state  $\uparrow_{i, \Phi}$  or  $\downarrow_\Phi$  at  $u$ , and the label of  $u$  (which can of course be computed by a test  $\text{lab}_f(x_1)$ ),  $M_\psi$  can determine the set  $Th_s(a, u)$ . If  $\psi$  is in that set (or more precisely, if  $\hat{\psi} \in Th_s(a, u)$ ), then  $u = b$  and  $M_\psi$  halts in state  $q_f$ . Now assume that it is not. Then  $M_\psi$  should determine, deterministically, where to go next. Suppose first that  $M_\psi$  is in state  $\uparrow_{i, \Phi}$ . If for some  $j \neq i$  the formula  $\psi_j(x_1, x_2)$ , defined as  $\exists z, y[\text{son}_j(x_2, z) \wedge y \leq z \wedge \psi(x_1, y)]$ , is in  $Th_s(a, u)$ , then  $u = \text{lca}(a, b)$  and  $M_\psi$  moves down to the  $j$ -th son of  $u$ ; otherwise,  $M_\psi$  moves up to the father of  $u$ . In the former case the new state is  $\downarrow_{\Phi'}$  where  $\Phi'$  can be determined (according to Section 8.2.2) from  $\Phi$ ,  $Th_s^\uparrow(u)$ , and all  $Th_s^\downarrow(u_m)$  with  $m \neq i, j$ . In the latter case the new state is  $\uparrow_{i', \Phi'}$  where  $i'$  can be checked by the test  $\exists y. \text{son}_{i'}(y, x_1)$ , and  $\Phi'$  can be determined from  $\Phi$  and all  $Th_s^\downarrow(u_j)$  with  $j \neq i$ . Suppose now that  $M_\psi$

is in state  $\downarrow_{\Phi}$ . This case is similar to, but easier than the previous one: if the formula  $\psi_j(x_1, x_2)$  is in  $Th_s(a, u)$  (and one of them must be), then  $M_\psi$  moves down to the  $j$ -th son of  $u$  in the state  $\downarrow_{\Phi'}$ , where  $\Phi'$  can now be determined from  $\Phi$  and all  $Th_s^\downarrow(u_i)$  with  $i \neq j$ .

It remains to explain how  $M_\psi$  starts its walk, in state  $q_{in}$ . It first tests whether  $a = b$ , using the rule  $\langle q_{in}, \psi(x_1, x_1) \rangle \rightarrow \langle q_f, \text{stay} \rangle$ . If that is not the case, then  $M_\psi$  uses the test formulas  $\psi_j(x_1, x_1)$ , i.e.,  $\exists z, y[\text{son}_j(x_1, z) \wedge y \leq z \wedge \psi(x_1, y)]$ . If such a formula is true, then  $M_\psi$  moves down to the  $j$ -th son of  $a$  in state  $\downarrow_{\Phi}$ , where  $\Phi$  can be determined from  $Th_s^\uparrow(a, a) = \{\varphi(x_1, x_2) \mid \varphi(x_1, x_1) \in Th_s^\downarrow(a)\}$  and all  $Th_s^\downarrow(a_i)$  with  $i \neq j$  (where  $a_i$  is the  $i$ -th son of  $a$ ). Otherwise,  $M_\psi$  moves up to the father of  $a$  in state  $\uparrow_{i, \Phi}$  where  $a$  is the  $i$ -th son of its father and  $\Phi = Th_s^\downarrow(a, a) = \{\varphi(x_1, x_2) \mid \varphi(x_1, x_1) \in Th_s^\downarrow(a)\}$ . Note that  $Th_s^\uparrow(a)$  and  $Th_s^\downarrow(a)$  can be computed by a test.

This ends the description of the nonjumping mstwt  $M_\psi$ . Due to the condition on  $\psi$ , which says that for every  $a$  there is a unique  $b$  such that  $\lfloor s \rfloor \models \psi(a, b)$ , the mstwt  $M_\psi$  is deterministic. ■

## 8.6 From global to local tests

We first show that MS tests can be simulated by tree-walking tree transducers in the special case where the input signature is monadic. For this we need the fact that a two-way automaton can keep track of the state of a deterministic one-way automaton. This is expressed in the next lemma, in an informal, but hopefully clear way.

**Lemma 8.10** Let  $M$  be an mstwt with a monadic input signature  $F$ , and let  $\mathcal{A}$  be a complete and deterministic finite  $F$ -automaton. Then an mstwt  $M'$  can be constructed that simulates  $M$  and keeps track of the state of  $\mathcal{A}$ ; in its non-local rules,  $M'$  uses the same MS formulas as  $M$ . The same holds if  $\mathcal{A}$  is top-down complete and deterministic. Determinism and the single-use restriction are preserved.

**Proof:** Let  $\mathcal{A} = \langle F, Q, \delta, Acc \rangle$  be a complete and deterministic finite automaton. The mstwt  $M'$  should simulate  $M$  and, for every input tree  $s \in T(F)$  and current node  $u \in N_s$ , keep track (in its finite state) of  $p = \text{state}_{s, \mathcal{A}}(u)$ . In fact,  $M'$  will behave in exactly the same way as  $M$ , except that after each simulation step it uses local epsilon-output rules to compute the new value of  $p$ , deterministically.

To initialize  $p$ ,  $M'$  first walks down to the unique leaf of  $s$  (its  $\epsilon$ -labelled node), and then walks back to the root, simulating  $\mathcal{A}$ ; note that it can recognize the root by the local test  $rt(x_1)$ . In this way  $M'$  computes  $p = \text{state}_{s, \mathcal{A}}(\text{root}_s)$ . Then it starts the simulation of  $M$ . Suppose that, simulating  $M$ ,  $M'$  moves up to the father of the current node. To update  $p$  it simply tests the label of the father, say  $f$ , and executes  $p := \delta_f(p)$ .

Suppose now that  $M'$  moves down to the son of the current node  $u$ . This is the difficult case. Let  $f$  be the label of  $u$ . If the label of the son  $u_1$  is  $\epsilon$ , then  $M'$  simply executes  $p := \delta_\epsilon$ . If it is not, then the new  $p$  must be in the set  $C = \{q \in Q \mid \delta_f(q) = p\}$  (where  $C$  stands for “candidates”). If  $C$  is a singleton the problem is solved. Now suppose it is not a singleton. Then  $M'$  walks down from  $u_1$  and computes for each descendant  $u'$  of  $u_1$ , and each  $q \in C$ , the set  $C(q, u')$  of all states  $q' \in Q$  such that, when started at  $u'$  in state  $q'$ ,  $\mathcal{A}$  reaches  $u_1$  in state  $q$ . Note that, for fixed  $u'$ , the sets  $C(q, u')$  are mutually disjoint. It is easy to see that  $M'$  can compute these sets, just by testing the labels of the nodes and using  $\delta$ . In fact,  $C(q, u_1) = \{q\}$  and if  $u''$  is the son of  $u'$ , then  $C(q, u'') = \{q'' \in Q \mid \delta_f(q'') \in C(q, u')\}$ , where  $f$  is the label of  $u'$ .

Now there are two cases: either  $M'$  arrives at a descendant  $u'$  such that exactly one of the sets  $C(q, u')$ , say  $C(\bar{q}, u')$ , is nonempty, whereas all the others are empty, or  $M'$  arrives at the leaf  $u'$  of  $s$ , in which case there is a unique  $\bar{q}$  such that  $\delta_\epsilon \in C(\bar{q}, u')$ . In both cases,  $M'$  knows that  $\bar{q} = \text{state}_{s, \mathcal{A}}(u_1)$  and executes  $p := \bar{q}$ . But now  $M'$  should return to  $u_1$ . To this purpose,  $M'$  moves up to the father  $u''$  of  $u'$  and picks two states  $q_1, q_2 \in Q$  that are in two distinct sets  $C(q, u'')$  (obviously, when walking down,  $M'$  can also store the candidate sets of the father nodes). Now  $M'$  starts up two copies  $\mathcal{A}_1$  and  $\mathcal{A}_2$  of the automaton  $\mathcal{A}$ , one in state  $q_1$  at  $u''$  and the other in state  $q_2$ . Then  $M'$  walks up, simulating both  $\mathcal{A}_1$  and  $\mathcal{A}_2$ . At the very moment where  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are in the same state (which is the old value of  $p$ ),  $M'$  is back on node  $u$ , and moves down to  $u_1$ . This ends the description of  $M'$ .

Finally, let  $\mathcal{A}$  be top-down complete and deterministic, with  $\text{Acc} = \{q_0\}$ . This time,  $M'$  should keep track of  $p = \text{tstate}_{s, \mathcal{A}}(u)$ . The proof is basically the same as the one above, interchanging up and down, and interchanging the roles of the root and the leaf of  $s$ . Initially,  $M'$  just executes  $p := q_0$ . When  $M$  moves down from  $u$ ,  $M'$  executes  $p := q$  where  $f[q] \rightarrow p$  is the unique rule in  $\delta$  with right-hand side  $p$  and  $f$  is the label of  $u$ . Now the difficult case is when  $M$  moves up to  $u$ . The computation of the new  $p$  is similar to the one above, with  $M'$  first walking up and computing the candidate sets for the ancestors of  $u$ , possibly reaching the root, then walking down to  $u_1$  and finally moving up to  $u$ . ■

**Theorem 8.11** For every mstwt with a monadic input signature an equivalent local mstwt can be constructed. Determinism and the single-use restriction are preserved.

**Proof:** Let  $M$  be an mstwt with monadic input signature  $F$ . By Theorem 8.9 we may assume that  $M$  is nonjumping. The remaining problem is how to simulate a global test of  $M$ , i.e., a formula  $\varphi \in \text{MS}(\mathcal{R}_F, \{x_1\})$  in the left-hand side of a rule of  $M$ , by an ordinary walk that uses local tests only.

So, let  $\varphi \in \text{MS}(\mathcal{R}_F, \{x_1\})$ , and let  $q$  be the quantifier-height of  $\varphi$ . It suffices to prove that  $M$  can be changed into an mstwt  $M'$  that simulates  $M$  and, moreover, for input tree  $s \in T(F)$  and current node  $u \in N_s$ , additionally keeps track (in its finite state) of  $p_1 = \text{Th}_s^\downarrow(u_1)$  (if  $u$  has a son  $u_1$ ) and of

$P = Th_s^\uparrow(u)$ . Recall from Section 8.2.2 that there is a mapping  $Z_f$  such that  $Th_s(u) = Z_f(Th_s^\uparrow(u), Th_s^\downarrow(u_1), \dots, Th_s^\downarrow(u_k))$ , where  $u$  has sons  $u_1, \dots, u_k$  (with  $k = 1$  or  $k = 0$ ) and label  $f$ . Hence, in  $M'$ , the global test  $\varphi$  can be replaced by the local test consisting of the disjunction of all formulas  $lab_f(x_1)$  such that  $\rho(f) = 1$  and  $\hat{\varphi} \in Z_f(P, p_1)$ , and the formula  $lab_\epsilon(x_1)$  if  $\hat{\varphi} \in Z_\epsilon(P)$ .

The construction of  $M'$  is possible by Lemma 8.10. Recall from Section 8.2.2 that there is a complete and deterministic finite  $F$ -automaton  $\mathcal{A}$  such that for every  $s \in T(F)$  and  $u \in N_s$ ,  $state_{s,\mathcal{A}}(u) = Th_s^\downarrow(u)$ . It is easy to construct from  $\mathcal{A}$  a complete and deterministic finite  $F$ -automaton  $\mathcal{A}_1$  such that  $state_{s,\mathcal{A}_1}(u) = \langle lab_s(u), Th_s^\downarrow(u_1) \rangle$  if  $u$  has a son  $u_1$ . Thus, by Lemma 8.10, there is an mstwt  $M_1$  that simulates  $M$  and keeps track of  $Th_s^\downarrow(u_1)$ . It should also be clear from Section 8.2.2, and from the fact that  $F$  is monadic, that there is a top-down complete and deterministic finite  $F$ -automaton  $\mathcal{A}_2$  such that  $tstate_{s,\mathcal{A}_2}(u) = Th_s^\uparrow(u)$  for every  $s \in T(F)$  and  $u \in N_s$ . In fact,  $\mathcal{A}_2$  has the set of states  $\mathcal{P}(L'_1)$  and the unique initial state  $Th_w(\text{root}_w)$ , which equals  $Th_s^\uparrow(\text{root}_s)$  for every  $s$ ; the transition rules of  $\mathcal{A}_2$  are obtained from the mappings  $Z_{f,1} : \mathcal{P}(L'_1) \rightarrow \mathcal{P}(L'_1)$  discussed in Section 8.2.2. Again by Lemma 8.10, applied to  $M_1$  and  $\mathcal{A}_2$ , there is an mstwt  $M'$  that simulates  $M$  and keeps track of both  $Th_s^\downarrow(u_1)$  and  $Th_s^\uparrow(u)$ . ■

The next theorem is now immediate from Theorems 8.8 and 8.11. It shows that the monadic second-order transductions of words are exactly those computed by two-way finite-state transducers (see Examples 8.4 and 8.6). This generalizes the monadic case of Theorem 8.1 to transductions.

**Theorem 8.12** DMSOW = 2DGSM.

In general, it is not possible to “remove” the MS tests from an mstwt, not even when it is an MS tree-walking automaton.

**Theorem 8.13** DTWTW  $\subset$  DTWTW<sup>MS</sup>.

**Proof:** Obviously, every regular tree language  $L$  can be recognized by a deterministic MS tree-walking automaton. In fact, the automaton need not walk at all: if the MS-sentence  $\varphi$  expresses  $L$  (by Theorem 8.1), then the rule  $\langle q_{\text{in}}, \varphi \rangle \rightarrow \epsilon$  suffices. However, by a result of Bojańczyk and Colcombet [BojCol06], not every regular tree language can be recognized by a tree-walking automaton. The proof is ingenious, and would take too much space here. ■

Thus, to “remove” MS tests in general, a more powerful type of transducer is needed. One such transducer is obtained by extending the nonjumping mstwt with the feature of a (restricted) pushdown. As opposed to the usual pushdown automaton, this extended mstwt can only push a symbol on the pushdown when it moves down to a son, and it has to pop the top symbol off the pushdown when it moves to the father. In other words, the movements of the pushdown are synchronized with the movements of the reading head of the mstwt. Initially (at the root) the pushdown contains one symbol, and hence, at each moment,

the number of symbols on the pushdown equals the depth of the current node  $u$ , i.e., its number of ancestors. Intuitively, each cell of the pushdown corresponds to an ancestor of  $u$ , and its content provides information about that ancestor; in particular, the top of the pushdown corresponds to  $u$  itself. From this point of view, it is natural to view the pushdown “upside down”: with its bottom at the root and its top at the current node.

Formally, an *MS tree-walking pushdown tree transducer* (abbreviated *p-mstwt*) is a tuple  $M = (F, H, Q, q_{\text{in}}, \Gamma, \gamma_{\text{in}}, R)$ , where  $F, H, Q$ , and  $q_{\text{in}}$  are the same as for an mstwt,  $\Gamma$  is an alphabet of pushdown symbols,  $\gamma_{\text{in}}$  is the initial pushdown symbol, and  $R$  is a finite set of rules. A rule is of the form  $\langle q, \varphi, \gamma \rangle \rightarrow \zeta$  with  $q \in Q$ ,  $\varphi \in \text{MS}(\mathcal{R}_F, \{x_1\})$ ,  $\gamma \in \Gamma$ , and either  $\zeta \in I$  or  $\zeta = h(\zeta_1, \dots, \zeta_k)$  with  $h \in H_k$  and  $\zeta_1, \dots, \zeta_k \in I$ , where  $I$  is the set of all triples  $\langle q', \psi, \beta \rangle$  with  $q' \in Q$ ,  $\psi \in \text{MS}(\mathcal{R}_F, \{x_1, x_2\})$ , and  $\beta \in \Gamma^*$ , and one of the following three cases holds:

- (a)  $\psi = \text{up}$ ,  $\beta = \varepsilon$ , and  
 $[s] \models \forall x_1 [\varphi(x_1) \Rightarrow \exists y. \text{son}(y, x_1)]$  for every  $s \in T(F)$ , or
- (b)  $\psi = \text{stay}$  and  $\beta \in \Gamma$ , or
- (c) there exists  $i \in [\rho(F)]$  such that  $\psi = \text{down}_i$ ,  $\beta \in \Gamma^2$ , and  
 $[s] \models \forall x_1 [\varphi(x_1) \Rightarrow \exists y. \text{son}_i(x_1, y)]$  for every  $s \in T(F)$ .

The p-mstwt  $M$  is *deterministic* if for every two distinct rules  $\langle q, \varphi_1, \gamma \rangle \rightarrow \zeta_1$  and  $\langle q, \varphi_2, \gamma \rangle \rightarrow \zeta_2$  in  $R$ ,  $\varphi_1$  and  $\varphi_2$  are mutually exclusive on  $T(F)$ .

Just as we did for an mstwt, we define for every input term  $s \in T(F)$  a regular grammar  $G_{M,s} = (H, X_{M,s}, R_{M,s})$ . The set  $X_{M,s}$  of configurations of  $M$  on  $s$  now consists of all triples  $\langle q, u, \pi \rangle$  with  $q \in Q$ ,  $u \in N_s$ , and  $\pi \in \Gamma^*$  such that the length of  $\pi$  equals the depth of  $u$ . If  $\langle q, \varphi, \gamma \rangle \rightarrow \zeta$  is a rule in  $R$ , then  $R_{M,s}$  contains all rules  $\langle q, u, \pi\gamma \rangle \rightarrow \zeta'$  for every  $u \in N_s$  such that  $[s] \models \varphi(u)$  and every  $\pi \in \Gamma^*$  with  $|\pi\gamma|$  equal to the depth of  $u$ , where  $\zeta'$  equals

1.  $\langle q', v, \pi\beta \rangle$ , and  $v$  is the unique node such that  $[s] \models \psi(u, v)$ ,  
if  $\zeta = \langle q', \psi, \beta \rangle$ ,
2.  $h(\langle q_1, v_1, \pi\beta_1 \rangle, \dots, \langle q_k, v_k, \pi\beta_k \rangle)$ , and  
 $v_i$  is the unique node such that  $[s] \models \psi_i(u, v_i)$  for every  $i \in [k]$ ,  
if  $\zeta = h(\langle q_1, \psi_1, \beta_1 \rangle, \dots, \langle q_k, \psi_k, \beta_k \rangle)$ .

The transduction computed by  $M$  is  $\tau_M = \{(s, t) \in T(F) \times T(H) \mid t \in L(G_{M,s}, \langle q_{\text{in}}, \text{root}_s, \gamma_{\text{in}} \rangle)\}$ .

Note that, by Theorem 8.9, for every mstwt there is an equivalent p-mstwt (with  $\Gamma = \{\gamma_{\text{in}}\}$ ).

As for an mstwt, a p-mstwt is *tree-to-word* if its output signature is monadic. Locality of a p-mstwt is defined in the same way as for an mstwt. By P-DTWT we denote the class of transductions computed by local deterministic p-mstwts, and by P-DTWTW the corresponding class in the tree-to-word case.

**Theorem 8.14** For every p-mstwt an equivalent local p-mstwt can be constructed. Determinism is preserved.

**Proof:** Let  $M$  be a p-mstwt with input signature  $F$ . As in Theorem 8.11, the problem is how to simulate a global test of  $M$ , i.e., a formula in the left-hand side of a rule of  $M$ , by an ordinary walk that uses local tests only. Let  $\varphi \in \text{MS}(\mathcal{R}_F, \{x_1\})$  be such a test, and let  $q$  be the quantifier-height of  $\varphi$ . As in the proof of Theorem 8.11, it suffices to prove that  $M$  can be changed into a p-mstwt  $M'$  that simulates  $M$  and, moreover, for input tree  $s \in T(F)$  and current node  $u \in N_s$ , additionally keeps track of  $p_1 = Th_s^\downarrow(u_1), \dots, p_k = Th_s^\downarrow(u_k)$  (if  $u$  has  $k$  sons  $u_1, \dots, u_k$ ) and of  $P = Th_s^\uparrow(u)$ .

We first show how  $M'$  can compute  $p_1, \dots, p_k$  (which are recomputed each time  $M'$  visits  $u$ ). Recall again from Section 8.2.2 that there is a complete and deterministic finite automaton  $\mathcal{A} = \langle F, Q, \delta, \text{Acc} \rangle$  such that for every  $s \in T(F)$  and  $u \in N_s$ ,  $\text{state}_{s, \mathcal{A}}(u) = Th_s^\downarrow(u)$ . Thus,  $p_i = \text{state}_{s, \mathcal{A}}(u_i)$  for every  $i \in [k]$ . Note that the definition of  $\text{state}_{s, \mathcal{A}}(u_i)$  is recursive; thus, to compute  $p_i$ ,  $M'$  implements this recursion on its pushdown in the obvious way: it executes a depth-first search of the subtree with root  $u_i$  and computes  $\text{state}_{s, \mathcal{A}}(u')$  for every descendant  $u'$  of  $u_i$ , in a bottom-up fashion. To this end, it uses additional pushdown symbols of the form  $(b, q_1 \cdots q_n)$  with  $b \in \{0, 1\}$  and  $q_1, \dots, q_n \in Q$ . The boolean  $b$  just indicates whether or not the corresponding node  $u'$  is  $u_i$  (this is because  $M'$  needs to know when it has returned to  $u_i$ ). The states  $q_1, \dots, q_n$  are the states of  $\mathcal{A}$  in which it reaches the first  $n$  sons of  $u'$ . Thus, after having computed  $p_1, \dots, p_{i-1}$  (and having stored them in its finite state),  $M'$  calls a subroutine that moves down to the  $i$ -th son of  $u$  in state  $\downarrow$  and pushes the symbol  $(1, \varepsilon)$ , where  $\varepsilon$  is the empty sequence of states of  $\mathcal{A}$ . The subroutine then executes the following rules, for every  $f \in F$  with  $\rho(f) = k$ ,  $b \in \{0, 1\}$ , and  $q_1, \dots, q_n \in Q$  with  $n \leq k$  (using the states  $\downarrow$  and  $\uparrow_{b,q}$  for every  $b \in \{0, 1\}$  and  $q \in Q$ ):

(1) if  $n < k$ , then

$$\langle \downarrow, \text{lab}_f(x_1), (b, q_1 \cdots q_n) \rangle \rightarrow \langle \downarrow, \text{down}_{n+1}, (b, q_1 \cdots q_n)(0, \varepsilon) \rangle,$$

(2) if  $n = k$  and  $\delta_f(q_1, \dots, q_n) = q$ , then

$$\langle \downarrow, \text{lab}_f(x_1), (b, q_1 \cdots q_n) \rangle \rightarrow \langle \uparrow_{b,q}, \varepsilon \rangle,$$

(3) if  $n < k$ , then

$$\langle \uparrow_{0,q}, \text{lab}_f(x_1), (b, q_1 \cdots q_n) \rangle \rightarrow \langle \downarrow, \text{stay}, (b, q_1 \cdots q_n q) \rangle.$$

Thus, the subroutine returns to  $u$  in some state  $\uparrow_{1,q}$ , and then  $M'$  stores  $p_i = q$  in its finite state.

It remains to show how  $M'$  keeps track of  $P = Th_s^\uparrow(u)$ . The value of  $P$  is stored on the pushdown. Thus, if  $M$  has pushdown alphabet  $\Gamma$  and initial pushdown symbol  $\gamma_{\text{in}}$ , then  $M'$  uses the pushdown symbols  $(\gamma, S)$  with  $\gamma \in \Gamma$  and  $S \subseteq L'_1$ , and has initial pushdown symbol  $(\gamma_{\text{in}}, Th_w(\text{root}_w))$ . Clearly, it suffices to show how  $P$  can be computed when  $M'$  moves down from node  $u$  to the  $i$ -th son  $u_i$ , simulating a move of  $M$ . That is easy: at  $u$ ,  $M'$  has

computed  $p_1 = Th_s^\downarrow(u_1), \dots, p_k = Th_s^\downarrow(u_k)$  (as indicated above), and it has stored  $P = Th_s^\uparrow(u)$  in the top of the pushdown. From this it can determine  $Th_s^\uparrow(u_i) = Z_{f,i}(P, p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_k)$  as observed in Section 8.2.2, and push  $Th_s^\uparrow(u_i)$  as second component of the new top symbol when moving down to  $u_i$ . ■

Theorems 8.8, 8.9, and 8.14 together show that the monadic second-order transductions of terms can be computed by tree-walking pushdown tree transducers.

**Theorem 8.15**  $DMSOT \subseteq P\text{-DTWT}$  and  $DMSOTW \subseteq P\text{-DTWTW}$ .

Using the construction in the proof of Theorem 8.14 it can now be shown that parameterless MS-transductions of terms and words can be implemented efficiently (but usually with large constants), cf. Proposition ??.

**Theorem 8.16** Every MS-transduction  $\tau$  in DMSOT can be computed in linear time, that is, for a given input term  $s$  the value of  $\tau(s)$  (or its nonexistence) can be computed in time  $O(|s|)$ .

**Proof:** Let  $\tau$  be an MS-transduction in DMSOT. By Theorems 8.8 and 8.9,  $\tau$  is computed by a single-use deterministic nonjumping mstwt  $M = (F, H, Q, q_{in}, R)$ . We first observe that  $M$  computes in linear time (assuming that each computation step takes one time unit). More precisely, for every  $s \in T(F)$  and  $t \in T(H)$ , if  $\langle q_0, root_s \rangle = \xi_0 \Rightarrow_{G_{M,s}} \xi_1 \Rightarrow_{G_{M,s}} \xi_2 \Rightarrow_{G_{M,s}} \dots \Rightarrow_{G_{M,s}} \xi_n = t$ , then  $n \leq |Q| \cdot |s|$ . To see this, we show that each configuration  $\langle q, u \rangle$  of  $M$  is rewritten at most once during this computation. Suppose it is rewritten both in  $\xi_i$  and in  $\xi_j$ , with  $i < j$ . Let  $\xi_i$  be of the form  $\xi_i = w_0 \langle q_1, u_1 \rangle w_1 \dots \langle q_k, u_k \rangle w_k$ , where each  $\langle q_l, u_l \rangle$  is a configuration and  $w_0, w_1, \dots, w_k$  do not contain configurations. Then  $\xi_j$  is of the form  $\xi_j = w_0 \zeta_1 w_1 \dots \zeta_k w_k$  with  $\langle q_l, u_l \rangle \Rightarrow_{G_{M,s}}^* \zeta_l$  for every  $l \in [k]$ . Let  $\langle q, u \rangle = \langle q_m, u_m \rangle$ . Then  $\langle q_m, u_m \rangle \neq \zeta_m$ , because  $\langle q, u \rangle$  is rewritten in  $\xi_i$ . Clearly,  $\langle q, u \rangle$  does not occur in  $\zeta_m$ , because otherwise the deterministic transducer  $M$  would repeat itself and the computation would be infinite. Thus,  $\langle q, u \rangle$  occurs in one of  $\zeta_1, \dots, \zeta_{m-1}, \zeta_{m+1}, \dots, \zeta_k$ . Consider the computation  $\xi_0 \Rightarrow_{G_{M,s}}^* \xi_i \Rightarrow_{G_{M,s}}^* \xi \Rightarrow_{G_{M,s}}^* \xi_j \Rightarrow_{G_{M,s}}^* t$  with  $\xi = w_0 \zeta_1 w_1 \dots \zeta_{m-1} w_{m-1} \langle q_m, u_m \rangle w_m \zeta_{m+1} w_{m+1} \dots \zeta_k w_k$ . It contradicts the fact that  $M$  is single-use, because  $\langle q, u \rangle$  occurs at least twice in  $\xi$ .

Thus, assuming that each computation step takes constant time,  $\tau$  can be computed in linear time: on input  $s$ , at most  $|Q| \cdot |s|$  computation steps of  $M$  are executed. However, in each computation step, global tests must be simulated. For each such test (with quantifier-height  $q$ ), as in the proof of Theorem 8.14, it now suffices to show that  $M$  can be simulated in such a way that, for input tree  $s \in T(F)$  and current node  $u \in N_s$ , the information  $p_1 = state_{s, \mathcal{A}}(u_1), \dots, p_k = state_{s, \mathcal{A}}(u_k)$  (if  $u$  has  $k$  sons  $u_1, \dots, u_k$ ) and  $P = Th_s^\uparrow(u)$  is available, where  $\mathcal{A}$  is the complete and deterministic  $F$ -automaton such that  $state_{s, \mathcal{A}}(u_i) = Th_s^\downarrow(u_i)$  (see Section 8.2.2).

Rather than computing the states  $p_1, \dots, p_k$  each time  $M$  visits  $u$  (as in the proof of Theorem 8.14), we preprocess the input tree  $s$  by first computing the (unique) run  $run_{\mathcal{A},s} : Pos(s) \rightarrow Q_{\mathcal{A}}$  of  $\mathcal{A}$  on  $s$ , where  $Q_{\mathcal{A}}$  is the set of states of  $\mathcal{A}$ . Recall that  $run_{\mathcal{A},s}(u) = state_{s,\mathcal{A}}(u)$  for every node  $u$  of  $s$ . Now  $p_1, \dots, p_k$  are available to  $M$  in constant time whenever it visits  $u$ . Obviously, the run of  $\mathcal{A}$  on  $s$  can be computed in time  $O(|s|)$ . And since  $M$  has only finitely many global tests, the total preprocessing time is linear.

The set of formulas  $P$  can be computed using the pushdown of a p-mstwt. As described at the end of the proof of Theorem 8.14, this pushdown can be updated in constant time in each computation step of  $M$  (using  $p_1, \dots, p_k$ ). Thus,  $P$  is available whenever  $M$  visits  $u$ . ■

Next we use Theorem 8.15 to prove that every mstwt can be simulated by a composition of two twts, the first of which only walks down in the input tree. A deterministic tree-walking tree transducer is called a *deterministic top-down tree transducer* if it does not use the formula up in the right-hand sides of its rules; the class of transductions computed by deterministic top-down tree transducers is denoted  $DTWT_{\downarrow}$ .

**Theorem 8.17** For every local p-mstwt  $M$  two twts  $M_1$  and  $M_2$  can be constructed such that  $\tau_M = \tau_{M_2} \circ \tau_{M_1}$ . Moreover,  $M_1$  is a deterministic top-down tree transducer, and if  $M$  is deterministic then so is  $M_2$ .

**Proof:** Let  $M = (F, H, Q, q_{in}, \Gamma, \gamma_{in}, R)$  be a local p-mstwt with  $\Gamma = \{\gamma_1, \dots, \gamma_n\}$ . Without loss of generality we assume that during the computations of  $M$ , the bottom symbol of the pushdown is always equal to  $\gamma_{in}$  ( $M$  can store the actual value of the bottom symbol in its finite state). We also assume w.l.o.g. that in point (c) of the definition of a p-mstwt,  $\beta = \gamma\gamma'$  for some  $\gamma' \in \Gamma$ , i.e., the symbol  $\gamma'$  is pushed on top of the pushdown, without changing the old top (if the old top has to be changed, first use a stay-rule). Finally, we assume that  $\beta = \gamma$  in point (b) of the definition of a p-mstwt, i.e., the pushdown is unchanged (if it has to be changed, first move up and then move down again).

The twt  $M_1$  preprocesses the input tree  $s$  in such a way that  $\tau_{M_1}(s)$  can be used by the twt  $M_2$  to simulate  $M$ , using the structure of  $\tau_{M_1}(s)$  to implement  $M$ 's pushdown. More precisely, for each node  $u \in N_s$  and each pushdown  $\pi \in \Gamma^m$ , where  $m$  is the depth of  $u$ ,  $\tau_{M_1}(s)$  has a node  $[u, \pi]$  with label  $[f, \gamma]$ , such that  $f$  is the label of  $u$  in  $s$  and  $\gamma$  is the top symbol of  $\pi$ . Thus,  $\pi$  corresponds to the second parts of the labels of the ancestors of  $[u, \pi]$ .

We define  $M_1 = (F, F \times \Gamma, \Gamma, \gamma_{in}, R_1)$ . Note that  $M_1$  uses the pushdown symbols of  $M$  as states. For every  $\gamma \in \Gamma$  and  $f \in F$  of arity  $k$ , the arity of  $[f, \gamma]$  is  $nk$ , and  $R_1$  contains the rule  $\langle \gamma, lab_f(x_1) \rangle \rightarrow \zeta_{\gamma,f}$ , with  $\zeta_{\gamma,f} =$

$$[f, \gamma](\langle \gamma_1, down_1 \rangle, \dots, \langle \gamma_n, down_1 \rangle, \dots, \langle \gamma_1, down_k \rangle, \dots, \langle \gamma_n, down_k \rangle).$$

This rule changes each son  $u_i$  of the current node  $u$  into  $n$  sons that receive the additional labels  $\gamma_1, \dots, \gamma_n$  respectively. Note that after application of the rule, the  $j$ -th copy of  $u_i$  is the  $m$ -th son of the output node, where  $m = n(i-1) + j$ .

Finally we define  $M_2 = (F \times \Gamma, H, Q, q_{\text{in}}, R_2)$  as follows. Consider a rule of  $M$ :  $\langle q, \varphi, \gamma \rangle \rightarrow \zeta$ . It is simulated by the rule

$$\langle q, \varphi' \wedge \bigvee_{f \in F} \text{lab}_{[f, \gamma]}(x_1) \rangle \rightarrow \zeta'$$

in  $R_2$ , where  $\varphi'$  is obtained from  $\varphi$  by changing every occurrence of an atomic formula  $\text{lab}_f(x_1)$  into the formula  $\bigvee_{\gamma \in \Gamma} \text{lab}_{[f, \gamma]}(x_1)$ , and  $\zeta'$  is obtained from  $\zeta$  by changing every triple  $\langle q', \text{up}, \varepsilon \rangle$  into  $\langle q', \text{up} \rangle$ , every triple  $\langle q', \text{stay}, \gamma \rangle$  into  $\langle q', \text{stay} \rangle$ , and every triple  $\langle q', \text{down}_i, \gamma \gamma_j \rangle$  into  $\langle q', \text{down}_{n(i-1)+j} \rangle$ . ■

Thus,  $\text{P-DTWT} \subseteq \text{DTWT} \circ \text{DTWT}_{\downarrow}$ . It is straightforward to show that this is, in fact, an equality (see also Theorem 8.27).

From Theorems 8.15 and 8.17 we obtain the next result.

**Theorem 8.18**  $\text{DMSOT} \subseteq \text{DTWT} \circ \text{DTWT}_{\downarrow}$ .

This theorem can be strengthened to a characterization of DMSOT in terms of tree-walking tree transducers, which can be viewed as a generalization of Theorem 8.1 to transductions. We omit the complicated proof (see [EngMan03a], as explained in Section 8.9.2). Recall from Section 8.1 that, trivially, every MS-transduction is of linear size increase. It should be clear that  $\text{DTWT} \circ \text{DTWT}_{\downarrow}$  contains transductions that are *not* of linear size increase (even  $\text{DTWT}_{\downarrow}$  does, cf. Example 8.7). The theorem shows that DMSOT and  $\text{DTWT} \circ \text{DTWT}_{\downarrow}$  have the same expressive power with respect to transductions of linear size increase. In other words, the monadic second-order transductions of terms are exactly the transductions of linear size increase that are computed by the composition of two tree-walking tree transducers, the first of which is a top-down tree transducer.

**Theorem 8.19** A transduction  $\tau$  of terms is in DMSOT if and only if (1)  $\tau$  is in  $\text{DTWT} \circ \text{DTWT}_{\downarrow}$  and (2)  $\tau$  is of linear size increase.

It is also shown in [EngMan03a], by the same complicated proof, that it is decidable for  $\tau \in \text{DTWT} \circ \text{DTWT}_{\downarrow}$  (given as a composition of two twts) whether or not  $\tau$  is of linear size increase, i.e., whether or not it is in DMSOT; and if so, a definition scheme for  $\tau$  can be constructed.

## 8.7 Nondeterminism

We will indicate the classes of transductions computed by nondeterministic devices (in particular: definition schemes with parameters) by dropping the D from the notation for the deterministic devices.

We first note that the nondeterminism of MS definition schemes is different in nature from the nondeterminism of tree-walking transducers. Intuitively this is because in a definition scheme one global guess for all input nodes is made at the beginning (by assigning values to the parameters) whereas a transducer

makes a local guess whenever it visits an input node, possibly making different guesses at different visits to the same node. Since, for given input term  $s$ , each parameter has  $2^{|s|}$  possible values, every transduction  $\tau$  in MSOT has finite images, i.e.,  $\tau(s)$  is finite for every input  $s$  (which also follows from the fact that  $\tau$  is of linear size increase, cf. Section 8.1).

**Proposition 8.20** The classes MSOW and 2GSM are incomparable. Hence the classes MSOT and  $\text{TWT}_{\text{su}}^{\text{MS}}$  are incomparable.

**Proof:** Obviously, two-way finite-state transducers can compute transductions that do not have finite images, for instance, the one that translates the word  $a$  (consisting of one symbol  $a$ ) into the words  $a^n$  for all  $n \in \mathbb{N}$ . In fact, even one-way finite-state transducers can do that, see also Example 8.5. Hence, 2GSM is not included in MSOW.

To show that MSOW is not included in 2GSM, consider the transduction  $\tau$  of words from  $A^* = \{a\}^*$  to  $B^* = \{b, c, \#\}^*$  such that for every  $a^n$ ,  $n \in \mathbb{N}$ ,  $\tau(a^n) = \{w\#w \mid w \in \{b, c\}^*, |w| = n\}$ . It is straightforward to show that  $\tau \in \text{MSOT}$ : the definition scheme is 2-copying and has one parameter, which indicates for each input position whether it will be relabelled with  $b$  or  $c$ . The details are left to the reader. This transduction cannot be computed by a two-way finite-state transducer. Intuitively that is because the transducer would have to visit each input position twice, but, in general, makes a different choice ( $b$  or  $c$ ) during the two visits. Formally, assume that  $\tau$  is computed by such a transducer  $M$  with  $k$  states. Choose  $n$  such that  $2^n > k(n+1)$ . Consider the behavior of  $M$  on input  $\mu_A(a^n)$ , i.e., the monadic tree  $a(a(\dots a(\epsilon)\dots))$ . Clearly,  $M$  has  $k(n+1)$  configurations on this input tree. Now consider the configuration  $M$  is in when it has just produced the output node with label  $\#$ . As there are  $2^n$  possible output words  $w\#w$  for  $a^n$ , there exist two words  $w_1$  and  $w_2$  for which this configuration is the same. This means that, on input  $\mu_A(a^n)$ ,  $M$  can switch its computation of  $w_1\#w_1$  halfway to the computation of  $w_2\#w_2$ , thus producing output  $w_1\#w_2$ , a contradiction. ■

Since the nondeterminism of a definition scheme just consists of guessing the values of the parameters, it can easily be expressed as a (nondeterministic, total) relabelling of terms, cf. Proposition 7.15 and Remark 7.18(1). Here we define a *term relabelling*  $\pi : T(F) \rightarrow T(H)$  to be a binary relation  $\pi \subseteq T(F) \times T(H)$  specified by a mapping  $r : F \rightarrow \mathcal{P}(H)$  such that for every  $f \in F$ ,  $r(f) \neq \emptyset$  and  $\rho(h) = \rho(f)$  for every  $h \in r(f)$ . By definition,  $\pi(f(t_1, \dots, t_k)) = \{h(t'_1, \dots, t'_k) \mid h \in r(f), t'_i \in \pi(t_i)\}$ . We denote the class of all term relabellings by REL.

**Proposition 8.21**  $\text{MSOT} = \text{DMSOT} \circ \text{REL}$ .

**Proof:** Let  $\tau : T(F) \rightarrow T(H)$  be an MS-transduction of terms, and let  $\mathcal{D}$  be a definition scheme of type  $\mathcal{R}_F \rightarrow \mathcal{R}_H$  defining  $\tau$ , with parameters  $X_1, \dots, X_n$ .

Recall from Definition 6.18 that  $F^{(n)}$  is the signature  $F \times \{0, 1\}^n$ , where  $(f, w)$  (for  $f \in F$  and  $w \in \{0, 1\}^n$ ) has arity  $\rho(f)$ . Also, for every input term  $s \in T(F)$  and every assignment  $\gamma : \{X_1, \dots, X_n\} \rightarrow \mathcal{P}(N_s)$ ,  $s * \gamma$  is the term

in  $T(F^{(n)})$  that is obtained from  $s$  by changing the label  $f$  of each node  $u$  of  $s$  into  $(f, (w_1, \dots, w_n))$  with  $w_i = 1$  if and only if  $u \in \gamma(X_i)$ .

Now define  $\pi : T(F) \rightarrow T(F^{(n)})$  to be the term relabelling specified by  $r(f) = \{(f, w) \mid w \in \{0, 1\}^n\}$  for  $f \in F$ . Define  $\mathcal{D}'$  to be the parameterless definition scheme of type  $\mathcal{R}_{F^{(n)}} \rightarrow \mathcal{R}_H$  that is obtained from  $\mathcal{D}$  by changing each of its formulas  $\varphi$  into the formula  $\varphi'$ , where  $\varphi'$  is obtained from  $\varphi$  by changing every occurrence of an atomic formula  $x \in X_i$  into the formula  $\bigvee_{(f, (w_1, \dots, w_n)) \in F^{(n)}, w_i=1} \text{lab}_{(f, (w_1, \dots, w_n))}(x)$ , and every occurrence of  $\text{lab}_f(x)$  into  $\bigvee_{(w_1, \dots, w_n) \in \{0, 1\}^n} \text{lab}_{(f, (w_1, \dots, w_n))}(x)$ . Clearly, for every input term  $s \in T(F)$  and every assignment  $\gamma : \{X_1, \dots, X_n\} \rightarrow \mathcal{P}(N_s)$ ,  $\lfloor s \rfloor \models \varphi$  if and only if  $\lfloor s * \gamma \rfloor \models \varphi'$  (for a sentence  $\varphi$ , and similarly for formulas with free variables). Consequently,  $\hat{\mathcal{D}}'(\lfloor s * \gamma \rfloor) = \hat{\mathcal{D}}(\lfloor s \rfloor, \gamma)$  and thus  $\tau = \tau' \circ \pi$ , where  $\tau'$  is the MS-transduction defined by  $\mathcal{D}'$ . This shows that  $\text{MSOT} \subseteq \text{DMSOT} \circ \text{REL}$ . The other inclusion follows from the obvious fact that  $\text{REL} \subseteq \text{MSOT}$  and that the class MSOT is closed under composition. ■

In view of this proposition, the nondeterministic case is not as interesting as the deterministic case. The results for DMSOT can be turned into results for MSOT by “adding” REL. In particular,  $\text{MSOT} = \text{DTWT}_{\text{su}}^{\text{MS}} \circ \text{REL}$  by Theorem 8.8,  $\text{MSOT} \subseteq \text{P-DTWT} \circ \text{REL}$  by Theorem 8.15, and MSOT is the class of transductions in  $\text{DTWT} \circ \text{DTWT}_{\downarrow} \circ \text{REL}$  of linear size increase by Theorem 8.19 (because term relabellings preserve size, and  $r(f) \neq \emptyset$  for every  $f \in F$ ).

For MSOW a similar, but slightly more complicated version of Proposition 8.21 holds (see [EngHoo01, Theorem 19]). For every word transduction  $\tau$  in  $\text{DMSOW} \circ \text{REL}$ ,  $\tau(\varepsilon)$  is a singleton because the  $\epsilon$  in the monadic input signature is not relabelled, and so  $\text{DMSOW} \circ \text{REL}$  is a proper subset of MSOW. Thus, instead of REL a slightly larger class is needed.

## 8.8 VR-equational sets of terms and words

Having characterized the MS-transductions of terms, we can apply the Equationality Theorem (Theorem 7.32) to obtain an automata-theoretic characterization of the VR-equational sets of terms. A set of terms  $L \subseteq T(F)$  is defined to be *VR-equational* if  $\text{Syn}(L) = \{\text{Syn}(t) \mid t \in L\}$  is VR-equational. A similar definition holds for sets of words, viewed as monadic trees. Note that the syntactic tree  $\text{Syn}(t)$  of a term  $t \in T(F)$  is an  $(F, [\rho(F)])$ -labelled graph (see Definition 2.13), and that the Equationality Theorem also holds for  $(K, L)$ -labelled graphs, as observed at the end of Section 7.2.

Assuming any reasonable definition of the representation  $[G]$  of a  $(K, L)$ -labelled graph  $G$ , it should be obvious that the representations of  $t$  and  $\text{Syn}(t)$  are equivalent with respect to MS-transductions. This is stated without proof in the next lemma.

**Lemma 8.22** Let  $F$  be a functional signature. Then there exist two MS-transductions  $\tau_1$  and  $\tau_2$  such that  $\tau_1(\lfloor \text{Syn}(t) \rfloor) = \lfloor t \rfloor$  and  $\tau_2(\lfloor t \rfloor) = \lfloor \text{Syn}(t) \rfloor$  for every  $t \in T(F)$ .

By Corollary 7.33 the class of VR-equational sets of graphs is preserved under MS-transductions. From this and the above lemma, it follows that the class of VR-equational sets of terms is preserved under MS-transductions of terms. In fact, consider an MS-transduction  $\tau : T(F) \rightarrow T(H)$  and a VR-equational set of terms  $L \subseteq T(F)$ . Let  $\tau' = \tau_2 \circ \tau \circ \tau_1$ . By Theorem 7.7,  $\tau'$  is an MS-transduction. Since  $\tau'(\text{Syn}(L)) = \text{Syn}(\tau(L))$  by Lemma 8.22,  $\tau(L)$  is VR-equational by Corollary 7.33. In particular, the class of VR-equational sets of words is preserved under MS-transductions of words (see the discussion in the introduction of this chapter).

To characterize the VR-equational sets of terms and words, we first prove an additional characterization of the MS-transductions of terms. Let MSOT-REL denote the class of MS-relabellings of terms, i.e., the set of all transductions  $\tau : T(F) \rightarrow T(H)$  such that  $\{(\lfloor s \rfloor, \lfloor t \rfloor) \mid (s, t) \in \tau\}$  is an MS-relabelling from  $STR(\mathcal{R}_F)$  to  $STR(\mathcal{R}_H)$ , see Definition 7.16. As usual, DMSOT-REL denotes the restriction to parameterless definition schemes, i.e.,  $\text{DMSOT-REL} = \text{DMSOT} \cap \text{MSOT-REL}$ .

**Theorem 8.23**  $\text{DTWT}_{\text{su}}^{\text{MS}} = \text{DTWT}_{\text{su}} \circ \text{DMSOT-REL}$ .

**Proof:** To prove that  $\text{DTWT}_{\text{su}}^{\text{MS}} \subseteq \text{DTWT}_{\text{su}} \circ \text{DMSOT-REL}$ , consider a deterministic single-use mstwt  $M = (F, H, Q, q_{\text{in}}, R)$ . By Theorem 8.9 we may assume that  $M$  is nonjumping. Let  $\varphi_1, \dots, \varphi_n \in \text{MS}(\mathcal{R}_F, \{x_1\})$  be all formulas that are used in the left-hand sides of the rules of  $M$ . Recall the notation from Definition 6.18 (as in the proof of Proposition 8.21). We will define an MS-relabelling  $\tau$  from  $T(F)$  to  $T(F^{(n)})$  such that  $\tau(s) = s * \gamma_s$ , where  $\gamma_s$  is the assignment  $\{X_1, \dots, X_n\} \rightarrow \mathcal{P}(N_s)$  with  $\gamma_s(X_i) = \{u \in N_s \mid \lfloor s \rfloor \models \varphi_i(u)\}$  for every  $i \in [n]$ . Then  $M$  can be changed into a local mstwt  $M'$  that receives  $\tau(s)$  as input and simulates  $M$  on input  $s$ : in the left-hand sides of its rules, it uses the boolean information in the label  $(f, (w_1, \dots, w_n))$  of the current node instead of the tests  $\varphi_1, \dots, \varphi_n$ .

To ensure that  $M'$  is deterministic, we take  $\tau$  from  $T(F)$  to  $T(F')$ , where  $F'$  is the subset of  $F^{(n)}$  consisting of all  $(f, (w_1, \dots, w_n))$  such that for all  $i, j \in [n]$ , if  $\varphi_i$  and  $\varphi_j$  are mutually exclusive on  $T(F)$ , then  $w_i = 0$  or  $w_j = 0$ . The transduction  $\tau$  is defined by the noncopying parameterless definition scheme  $\mathcal{D} = \langle \chi, \delta, (\theta_R)_{R \in \mathcal{R}_{F'}} \rangle$  of type  $\mathcal{R}_F \rightarrow \mathcal{R}_{F'}$  where  $\chi = \text{True}$ ,  $\delta = \text{True}$ ,  $\theta_{\text{lab}_{(f, (w_1, \dots, w_n))}} = \text{lab}_f(x_1) \wedge (\bigwedge_{i \in [n], w_i=1} \varphi_i) \wedge (\bigwedge_{i \in [n], w_i=0} \neg \varphi_i)$ , and  $\theta_{\text{son}_i} = \text{son}_i(x_1, x_2)$ . Note the similarity with Definition 7.19 (labelling by theories). The twt  $M'$  is obtained from  $M$  by changing, in the left-hand sides of its rules, each  $\varphi_i$  into  $\bigvee_{(f, (w_1, \dots, w_n)) \in F', w_i=1} \text{lab}_{(f, (w_1, \dots, w_n))}(x_1)$ . Obviously, it is deterministic and single-use.

The inclusion in the other direction follows from Theorem 8.8 and the fact that DMSOT is closed under composition. ■

**Theorem 8.24**  $\text{DMSOT} = \text{DTWT}_{\text{su}} \circ \text{DMSOT-REL}$  and  
 $\text{MSOT} = \text{DTWT}_{\text{su}} \circ \text{MSOT-REL}$ .

**Proof:** The first equation is immediate from Theorems 8.8 and 8.23. The second equation then follows from Proposition 8.21 and the fact that the equation  $\text{DMSOT-REL} \circ \text{REL} = \text{MSOT-REL}$  is an easy special case of that same proposition. ■

Now we can characterize the VR-equational sets of terms and words.

**Theorem 8.25** A set of terms is VR-equational if and only if it is in  $\text{DTWT}_{\text{su}}(\text{REGT})$ .

**Proof:** Let  $L \subseteq T(F)$  be VR-equational. By the Equationality Theorem (Theorem 7.32), there is a functional signature  $H$  and an MS-transduction  $\tau$ , such that  $\text{Syn}(L) = \tau(T(H))$ . Since the class of MS-transductions is closed under composition (Theorem 7.7), Lemma 8.22 implies that there is an MS-transduction  $\tau'$  from  $T(H)$  to  $T(F)$  such that  $L = \tau'(T(H))$ . By Theorem 8.24 there is an MS-relabelling  $\rho$  and a transduction  $\mu \in \text{DTWT}_{\text{su}}$  such that  $L = \mu(\rho(T(H)))$ . Since  $T(H)$  is regular,  $\rho(T(H))$  is regular by Proposition 8.2(2). This shows that  $L \in \text{DTWT}_{\text{su}}(\text{REGT})$ .

Now let  $L = \mu(R)$  with  $\mu \in \text{DTWT}_{\text{su}}$  and  $R \in \text{REGT}$ , with  $L \subseteq T(F)$  and  $R \subseteq T(H)$ . By Theorem 8.8,  $\mu$  is in MSOT. Changing the formula  $\chi$  of the definition scheme into  $\chi \wedge \varphi$ , where  $\varphi$  MS-defines  $R$  (see Theorem 8.1), an MS-transduction  $\tau$  is obtained such that  $L = \tau(T(H))$ . As above, Theorem 7.7 and Lemma 8.22 imply that there is an MS-transduction  $\tau'$  such that  $\text{Syn}(L) = \tau'(T(H))$ . By the Equationality Theorem,  $L$  is VR-equational. ■

Since deterministic twts with a monadic output signature are single-use, this immediately gives the next result.

**Theorem 8.26** A set of words is VR-equational if and only if it is in  $\text{DTWTW}(\text{REGT})$ .

We note that in Theorems 8.25 and 8.26 one can replace REGT by the set of all  $T(H)$  where  $H$  is a functional signature, i.e., a set of terms is VR-equational if and only if it is the range of a transduction in  $\text{DTWT}_{\text{su}}$ , and similarly for words. That is because every regular tree language can be obtained from the set of parse trees of a context-free grammar by a deterministic term relabelling (specified by a mapping  $r$  such that  $r(f)$  is a singleton for every  $f$ ), and every such parse tree language can be recognized by a deterministic tree-walking automaton, by a depth-first search.

We also note that the class of context-free languages is properly contained in the class of VR-equational languages (see the discussion in the introduction of this chapter). To show the containment, let  $G$  be a context-free grammar. It is well known, and easy to see, that the set  $P(G)$  of parse trees of  $G$  is a regular tree language. Since  $L(G) = \text{yd}(P(G))$ , where  $L(G)$  is the language generated by  $G$ , and since the yield mapping  $\text{yd}$  is in  $\text{DTWTW}$  (see Example 8.3),  $L(G)$

is in DTWTW(REGT) and hence VR-equational by Theorem 8.26. Again by Theorem 8.26, the language  $\tau(L) = \{a^n b^n c^n d^n \mid n \geq 0\}$  of Example 8.4 is a noncontext-free VR-equational language.

We finally note that, by Theorem 5.64, every VR-equational set  $L$  of terms (or words) has a decidable MS-theory. As discussed in Section 7.5, this is equivalent (by Theorem 8.1) to the fact that it is decidable for a finite automaton  $\mathcal{A}$  whether  $L \cap L(\mathcal{A}) = \emptyset$ . The latter problem is, in fact, decidable for every  $L \in \text{TWT(REGT)}$  in exponential time (see [Eng09]).

## 8.9 Bibliographic remarks

It is shown in [EngHey94] that HR and VR context-free graph grammars have the same term and word generating power, because the graphs  $Syn(t)$  corresponding to terms  $t$  are of bounded degree. This also means that the VR-equational and HR-equational sets of terms and words are the same (with an appropriate definition of HR-equational sets of terms and words), see Theorem 4.45.

### 8.9.1 Words

Two-way finite-state transducers are basically just a particular restriction of two-tape Turing machines; they were first studied in [AhoUll70]. Lemma 8.10 is a general technique for two-way machines, proved in Lemma 3 of [HopUll67] (see also page 212 of [AHU69]). Using this lemma, it was shown in [ChyJák77] that the class 2DGSM is closed under composition.

The characterization  $\text{DMSOW} = \text{2DGSM}$  (Theorem 8.12) is the main result of [EngHoo01], where the hybrid MS two-way finite-state transducer was introduced as a technical tool. The incomparability of MSOW and 2GSM (Proposition 8.20) is Corollary 16 of [EngHoo01]. The relationship between MSOT and DMSOT of Proposition 8.21 is Theorem 18 of [EngHoo01], where it is stated for graphs; for the case of words see Theorem 19 of [EngHoo01]. It is shown in [EngHoo07] that a word transduction  $\tau$  is in MSOW if and only if (1)  $\tau$  is in the class  $\text{2DGSM} \circ \text{2GSM}$  and (2)  $\tau$  has finite images.<sup>1</sup> Moreover, it is decidable for  $\tau \in \text{2DGSM} \circ \text{2GSM}$  (given as a composition of two two-way finite-state transducers) whether or not  $\tau$  has finite images, i.e., whether or not it is in MSOW; and if so, a definition scheme for  $\tau$  can be constructed. And finally, these results even hold for transductions  $\tau$  that are compositions of arbitrarily many two-way finite-state transducers. It should also be noted that, in the above statements, ‘has finite images’ can be replaced by ‘is of linear size increase’.

A Hennie machine is a two-way finite-state transducer that can also write on its input tape, just as a Turing machine. However, it may only visit each cell of its input tape a bounded number of times. It is shown in [EngHoo01] that

---

<sup>1</sup>Since, obviously, every rational transduction is in 2GSM, this proves (and strengthens) the result mentioned at the end of Example 8.5: a rational transduction is an MS-transduction if and only if it has finite images.

MSOW (DMSOW) is the class of word transductions computed by nondeterministic (deterministic) Hennie machines.

The word generating power of HR graph grammars was first investigated in [HabKre87] and Chapter V of [Hab92], see also Section 2.5.2 of [DHK97]. The tree-walking tree-to-word transducer was introduced in [AhoUll71], as a model of syntax-directed translation. The characterization in Theorem 8.26 of the VR-equational sets of words by the class DTWTW(REGT) is the main result of [EngHey91] (with a direct proof, not involving MS logic). A characterization of DTWTW(REGT) in terms of a special type of deterministic top-down tree transducers (taking the yields of the output trees) can be found in Corollary 4.11 of [ERS80]; it is basically due to [AhoUll71]. A special case of the Semi-linearity Theorem (Theorem 7.37) is stated in Corollary 3.2.7 of [ERS80]. Still other characterizations of DTWTW(REGT) can be found in Section 6 of [Eng97]. A characterization of TWTW(REGT) as path languages of VR-equational sets of graphs is proved in Theorem 28 of [EngOos96].

A characterization of DMSOTW (i.e., of  $\text{DTWTW}^{\text{MS}}$ , see Theorem 8.8) in terms of deterministic top-down tree transducers (again, taking the yields of the output trees) is given in Theorem 7.7 of [EngMan99].

## 8.9.2 Terms

As observed in the previous subsection, the tree-walking tree-to-word transducer was introduced in [AhoUll71] as a model of syntax-directed translation. The tree-walking tree transducer is essentially a notational variation of the *attribute grammar*, a compiler construction tool [DJL88, WilMau95, ALSU07] introduced in [Knu68]. An attribute grammar has attributes instead of states, and the output tree is usually interpreted in some semantic domain. When viewed as a tree transducer, the attribute grammar is called an attributed tree transducer (see [Fül81, FülVog98]). The current version of the twt is introduced as a special case of the *pebble tree transducer* in [MSV03], viz. the case that the transducer has no pebbles (except for the reading head, which is also viewed as a pebble in [MSV03]); the pebble tree transducer is proposed in [MSV03] as a model of XML transformations. The relationship between the attribute grammar and the deterministic tree-walking tree transducer is explained in Section 3.2 of [EngMan03b] (where the twt is called 0-ptt). The deterministic twt is slightly more powerful than the attribute grammar, because the latter is assumed to be noncircular (i.e., always halting, in a strong sense). The single-use restriction was introduced (for attribute grammars) in [Gan83, Gie88], where it is shown that single-use attributed tree transducers are closed under composition.

The basic characterization  $\text{DMSOT} = \text{DTWT}_{\text{su}}^{\text{MS}}$  of Theorem 8.8 is one of the main results of [BloEng00]; more precisely, it is proved there, in terms of attribute grammars, that  $\text{DMSOT} = \text{DTWT}_{\text{su}} \circ \text{DMSOT-REL}$ , see Theorem 8.24. How to get rid of jumps (Theorem 8.9) is shown in Theorems 8 and 9 of [BloEng97]. The impossibility to get rid of global tests (Theorem 8.13) is proved in [BojCol06]. The class DMSOT-REL is characterized in terms of finite-valued attribute grammars in Theorem 3.7 of [NevBus02] and in Theo-

rem 10 of [BloEng00] (independently), and in terms of compositions of bottom-up and top-down finite-state relabellings in Theorem 4.4 of [EngMan99] (cf. the discussion following Lemma 4.5 of [EngMan99]). It implies that the deterministic MS-transductions of terms can be computed in linear time (Corollary 18 of [BloEng00]), see Theorem 8.16.

The tree-walking pushdown tree-to-word transducer was introduced in [ERS80] (where it is called the ct-pd transducer), and the tree-walking pushdown tree transducer was introduced in [EngVog86] (where it is called the RT(P(TR)) transducer or indexed tree transducer), cf. Theorem 8.15. The pushdown of a p-mstwt can also be viewed as a pushdown of (coloured) pebbles: the pebbles are placed on the ancestors of the current node. In [EHS07] the p-twt is generalized to a transducer called I-PTT, that can place the pebbles anywhere on the input tree; it also generalizes the pebble tree transducer of [MSV03]. The decomposition result of Theorem 8.17 is a special case of Lemma 3 of [EHS07] (where the special case allows the first transducer to be deterministic).

To appreciate the relationship of Theorems 8.15, 8.18, and 8.19 to the literature, it is necessary to discuss the *macro tree transducer* [Eng80, CouFra82, EngVog85]. The macro tree transducer is obtained from the top-down tree transducer by letting its states (which can be viewed as recursive function procedures) have parameters of type ‘output tree’; for a formal definition that is close to the twt, see [EngMan03b]. Let DMT denote the class of tree transductions computed by deterministic macro tree transducers, with unrestricted rewriting, or, equivalently, outside-in rewriting (it is denoted  $DMT_{OI}$  in [EngVog85]).

**Theorem 8.27**  $DMT = P-DTWT = DTWT \circ DTWT_{\downarrow}$ .

**Proof:** It is proved in Theorem 5.16 of [EngVog86] (with  $S = TR$ ) that for every total function  $\tau$ ,  $\tau \in DMT$  if and only if  $\tau \in P-DTWT$  (this is a transducer version of the well-known fact that macro grammars and indexed grammars have the same power). In Theorem 6.18 of [EngVog85] it is shown that every  $\tau \in DMT$  can be written as  $\tau = \tau_2 \circ \tau_1$ , where  $\tau_1$  is the identity on some regular set  $R$  of trees, and  $\tau_2 \in DMT$  is a total function and hence  $\tau_2 \in P-DTWT$ . It should be clear that  $\tau_2 \circ \tau_1$  is in P-DTWT: the p-twt just starts by checking that the input tree is in  $R$  (see the proof of Theorem 8.14). This proves that  $DMT \subseteq P-DTWT$ . The inclusion  $P-DTWT \subseteq DTWT \circ DTWT_{\downarrow}$  is shown in Theorem 8.17. Finally, the inclusion  $DTWT \circ DTWT_{\downarrow} \subseteq DMT$  follows from Theorem 35 of [EngMan03b] (which shows, for  $n = 0$ , that  $DTWT \subseteq DMT$ ) and Theorem 7.6(3) of [EngVog85] (which shows that  $DMT \circ DTWT_{\downarrow} \subseteq DMT$ ). ■

The inclusions of Theorems 8.15 and 8.18 are proved in [EngMan99] in the form  $DMSOT \subseteq DMT$ . The characterization in Theorem 8.19 of DMSOT as the class of transductions in  $DTWT \circ DTWT_{\downarrow}$  of linear size increase, is the main result of [EngMan03a], where it is proved for DMT rather than  $DTWT \circ DTWT_{\downarrow}$ . The proof is for total functions only, but it should be clear from (the proof of) Theorem 6.18 of [EngVog85] (cf. the above proof) that it then also holds for partial functions: note that if a partial function  $\tau$  is in DMSOT then it

can be written as  $\tau = \tau_2 \circ \tau_1$ , where  $\tau_1$  is the identity on some regular set of trees, and  $\tau_2 \in \text{DMSOT}$  is a total function (just view the formula  $\chi$  of the definition scheme for  $\tau$  as a separate transduction  $\tau_1$ , and take the definition scheme for  $\tau_2$  to be the one of  $\tau$  with  $\chi = \text{True}$ ). The decidability result mentioned after Theorem 8.19 is also proved in [EngMan03a]. It is shown in [Man03] that the if-direction of Theorem 8.19 even holds for transductions  $\tau$  that are compositions of arbitrarily many deterministic twts (or macro tree transducers). In Theorem 7.1 of [EngMan99] DMSOT is characterized by syntactically restricted classes of macro tree transducers.

No results for MSOT and TWT are known that are similar to those for MSOW and 2GSM. Since it is easy to see that  $\text{REL} \subseteq \text{TWT}_\downarrow$  (the top-down tree transducer visits each node exactly once and guesses its new label), it follows from Proposition 8.21 and Theorem 8.18 that MSOT is included in  $\text{DTWT} \circ \text{DTWT}_\downarrow \circ \text{TWT}_\downarrow$ . But it is unknown whether MSOT equals the set of transductions of linear size increase in this class.

The term generating power of HR graph grammars was first investigated in [EngHey92], where it is shown (by a direct proof, not involving logic) that  $\text{DTWT}(\text{REGT})$  is the class of sets of terms that can be generated by HR graph grammars, if one allows the graph grammars to generate directed graphs without circuit (i.e., trees with shared subtrees), cf. the discussion at the end of Section 8.4. The characterization in Theorem 8.25 of the VR-equational sets of terms by the class  $\text{DTWT}_{\text{su}}(\text{REGT})$  is stated in Corollary 19 of [BloEng00]. Two other, closely related, characterizations of this class are given in Theorem 8.1 of [Dre99] (in terms of top-down tree transducers and hypergraph substitution) and in Theorem 5 of [EngMan00] (in terms of macro tree transducers; see also Corollary 7.3 of [EngMan99]). In [EngMan00] several natural types of term generating HR graph grammars are investigated.

We finally note that the equivalence problem is decidable for transductions in DMSOT, as proved in [EngMan06]. This result extends to MS-transductions from graphs to terms on VR-equational sets of graphs, but there is no hope to extend it to all graphs. For words, i.e., for deterministic two-way finite-state transducers, the decidability of the equivalence problem was proved in [Gur82]. It is open whether it is decidable for deterministic tree-walking tree transducers (and hence for deterministic macro tree transducers, cf. [Eng80]).

# Bibliography

- [AHU69] A. V. Aho, J. E. Hopcroft, J. D. Ullman. A general theory of translation, *Mathematical Systems Theory* 3 (1969) 193–221
- [ALSU07] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman. *Compilers; Principles, Techniques, & Tools*, Pearson Education, Boston, 2007
- [AhoUll70] A. V. Aho, J. D. Ullman. A characterization of two-way deterministic classes of languages, *Journal of Computer and System Sciences* 4 (1970) 523–538
- [AhoUll71] A. V. Aho, J. D. Ullman. Translations on a context-free grammar, *Information and Control* 19 (1971) 439–475
- [AutBoa88] J.-M. Autebert, L. Boasson. *Transductions rationnelles: application aux langages algébriques*, Masson, Paris 11, 1988
- [Ber79] J. Berstel. *Transductions and Context-Free Languages*, Teubner Studienbücher, Stuttgart, 1979
- [BloEng97] R. Bloem, J. Engelfriet. Monadic second order logic and node relations on graphs and trees, in: *Structures in Logic and Computer Science* (J. Mycielski, G. Rozenberg, A. Salomaa, eds.), *Lecture Notes in Computer Science*, v. 1261, Springer-Verlag, 144–161, 1997
- [BloEng00] R. Bloem, J. Engelfriet. A comparison of tree transductions defined by monadic second order logic and by attribute grammars, *Journal of Computer and System Sciences* 61 (2000) 1–50
- [BojCol06] M. Bojańczyk, T. Colcombet. Tree-walking automata cannot be determinized, *Theoretical Computer Science* 350 (2006) 164–173
- [Büc60] J. R. Büchi. Weak second-order arithmetic and finite automata, *Z. Math. Logik Grundlagen Math.* 6 (1960) 66–92
- [ChyJák77] M. P. Chytil, V. Ják. Serial composition of 2-way finite-state transducers and simple programs on strings, in: *Proceedings 4th ICALP* (A. Salomaa, M. Steinby, eds.), *Lecture Notes in Computer Science*, v. 52, Springer Verlag, 135–147, 1977

- [CouFra82] B. Courcelle, P. Franchi-Zannettacci. Attribute grammars and recursive program schemes I,II, *Theoretical Computer Science* 17 (1982) 163–191, 235–257
- [CouMos93] B. Courcelle, M. Mosbah. Monadic second-order evaluations on tree decomposable graphs, *Theoretical Computer Science* 109 (1993) 49–82
- [Don70] J. Doner. Tree acceptors and some of their applications, *Journal of Computer and System Sciences* 4 (1970) 406–451
- [Dre99] F. Drewes. A characterization of the sets of hypertrees generated by hyperedge-replacement graph grammars, *Theory of Computing Systems* 32 (1999), 159–208
- [DHK97] F. Drewes, A. Habel, H.-J. Kreowski. Hyperedge replacement graph grammars, in: *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations* (G. Rozenberg, ed.), World Scientific, 95–162, 1997
- [DJL88] P. Deransart, M. Jourdan, B. Lorho. *Attribute Grammars*, Lecture Notes in Computer Science, v. 323, Springer-Verlag, 1988
- [Elg61] C. C. Elgot. Decision problems of finite automata design and related arithmetics, *Trans. AMS* 98 (1961) 21–52
- [Eng80] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages, in: *Formal Language Theory; Perspectives and Open Problems* (R. V. Book, ed.), Academic Press, New York, 1980
- [Eng97] J. Engelfriet. Context-free graph grammars, in: *Handbook of Formal Languages, Volume 3: Beyond Words* (G. Rozenberg, A. Salomaa, eds.), Springer-Verlag, 125–213, 1997
- [Eng09] J. Engelfriet. The time complexity of typechecking tree-walking tree transducers, *Acta Informatica* 46 (2009) 139–154
- [EngHey91] J. Engelfriet, L.M. Heyker. The string generating power of context-free hypergraph grammars, *Journal of Computer and System Sciences* 43 (1991) 328–360
- [EngHey92] J. Engelfriet, L.M. Heyker. Context-free hypergraph grammars have the same term-generating power as attribute grammars, *Acta Informatica* 29 (1992) 161–210
- [EngHey94] J. Engelfriet, L.M. Heyker. Hypergraph languages of bounded degree, *Journal of Computer and System Sciences* 48 (1994) 58–89

- [EngHoo01] J. Engelfriet, H.J. Hoogeboom. MSO definable string transductions and two-way finite state transducers, *ACM Transactions on Computational Logic* 2 (2001) 216–254
- [EngHoo07] J. Engelfriet, H.J. Hoogeboom. Finitary compositions of two-way finite-state transductions, *Fundamenta Informaticae* 80 (2007) 111–123
- [EHS07] J. Engelfriet, H.J. Hoogeboom, B. Samwel. XML transformation by tree-walking transducers with invisible pebbles, in: *Proceedings 26th PODS* (L. Libkin, ed.), June 2007, Beijing, ACM Press, pp.63–72
- [EngMan99] J. Engelfriet, S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations, *Information and Computation* 154 (1999) 34–91
- [EngMan00] J. Engelfriet, S. Maneth. Tree languages generated by context-free graph grammars, in: *Theory and Applications of Graph Transformations, 6th International Workshop, TAGT'98* (H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg, eds.), *Lecture Notes in Computer Science*, v. 1764, Springer Verlag, 15–29, 2000
- [EngMan03a] J. Engelfriet, S. Maneth. Macro tree translations of linear size increase are MSO definable, *SIAM Journal on Computing* 32 (2003) 950–1006
- [EngMan03b] J. Engelfriet, S. Maneth. A comparison of pebble tree transducers with macro tree transducers, *Acta Informatica* 39 (2003), 613–698
- [EngMan06] J. Engelfriet, S. Maneth. The equivalence problem for deterministic MSO tree transducers is decidable, *Information Processing Letters* 100 (2006) 206–212
- [EngOos96] J. Engelfriet, V. van Oostrom. Regular description of context-free graph-languages, *Journal of Computer and System Sciences* 53 (1996) 556–574
- [ERS80] J. Engelfriet, G. Rozenberg, G. Slutzki. Tree transducers, L systems, and two-way machines, *Journal of Computer and System Sciences* 20 (1980) 150–202
- [EngVog85] J. Engelfriet, H. Vogler. Macro tree transducers, *Journal of Computer and System Sciences* 31 (1985) 71–146
- [EngVog86] J. Engelfriet, H. Vogler. Pushdown machines for the macro tree transducer, *Theoretical Computer Science* 42 (1986) 251–368

- [Fül81] Z. Fülöp. On attributed tree transducers, *Acta Cybernetica* 5 (1981) 261–279
- [FülVog98] Z. Fülöp, H. Vogler. *Syntax-Directed Semantics–Formal Models based on Tree Transducers*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1998
- [Gan83] H. Ganzinger. Increasing modularity and language-independency in automatically generated compilers, *Science of Computer Programming* 3 (1983) 223–278
- [Gie88] R. Giegerich. Composition and evaluation of attribute coupled grammars, *Acta Informatica* 25 (1988), 355–423
- [Gur82] E. M. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable, *SIAM Journal on Computing* 11 (1982) 448–452
- [HabKre87] A. Habel, H.-J. Kreowski. Some structural aspects of hypergraph languages generated by hyperedge replacement, in: *Proceedings STACS’87, Lecture Notes in Computer Science*, v. 247, 207–219, 1987
- [Hab92] A. Habel. *Hyperedge Replacement: Grammars and Languages*, *Lecture Notes in Computer Science*, v. 643, 1992
- [HopUll67] J. E. Hopcroft, J. D. Ullman. An approach to a unified theory of automata, *The Bell System Technical Journal* 46 (1967) 1793–1829
- [Knu68] D. E. Knuth. Semantics of context-free languages, *Mathematical Systems Theory* 2 (1968) 127–145
- [Man03] S. Maneth. The macro tree transducer hierarchy collapses for functions of linear size increase, in: *Proceedings FSTTCS 2003*, 326–337
- [MSV03] T. Milo, D. Suci, V. Vianu. Typechecking for XML transformers, *Journal of Computer and System Sciences* 66 (2003) 66–97
- [NevBus02] F. Neven, J. Van den Bussche. Expressiveness of structured document query languages based on attribute grammars, *Journal of the ACM* 49 (2002) 56–100
- [Saka03] J. Sakarovitch. *Éléments de théorie des automates*, Vuibert, Paris, 2003 (to be published by Cambridge University Press in English)
- [ThaWri68] J. W. Thatcher, J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic, *Mathematical Systems Theory* 2 (1968) 57–82

- [Tra62] B. A. Trakhtenbrot. Finite automata and the logic of one-place predicates, *Siberian Mathematical Journal* 3 (1962) 103–131 (in Russian). English translation: American Mathematical Society Translations, Series 2, 59 (1966) 23–55
- [WilMau95] R. Wilhelm, D. Maurer. *Compiler Design*, Addison-Wesley, 1995

