Algebraic Description of Derivational Minimalism

Jens Michaelis and Uwe Mönnich and Frank Morawietz
Seminar für Sprachwissenschaft
Universität Tübingen
Wilhelmstr. 113, 72074 Tübingen
Germany
{michael,um,frank}@sfs.nphil.uni-tuebingen.de

Abstract

In this paper we extend the work by Michaelis (1999) which shows how to encode an arbitrary Minimalist Grammar in the sense of Stabler (1997) into a weakly equivalent multiple context-free grammar (MCFG). By viewing MCFG rules as terms in a free Lawvere theory we can translate a given MCFG into a regular tree grammar. The latter is characterizable by both a tree automaton and a corresponding formula in monadic second-order (MSO) logic. The trees of the resulting regular tree language are then unpacked into the intended “linguistic” trees with an MSO transduction based upon tree-walking automata. This two-step approach gives an operational as well as a logical description of the tree sets involved.

1 Introduction

Algebraic, logical and regular characterizations of (tree) languages provide a natural framework for the denotational and operational semantics of grammar formalisms relying on the use of trees for their intended models.

Over the last couple of years, a rich class of mildly context-sensitive grammar formalisms has been proven to be weakly equivalent. Among others, the following families of (string) languages are equivalent: $STR(HR)$ [languages generated by string generating hyperedge replacement grammars], $OUT(DTWT)$ [output languages of deterministic tree-walking tree-to-string transducers], $yDT_{fc}(REGT)$ [yields of images of regular tree languages under deterministic finite-copying top-down tree transductions], $MCFL$ [languages generated by multiple context-free grammars], $MCTAL$ [languages generated by multi-component tree adjoining grammars], $LCFRL$ [languages generated by linear context-free rewriting systems], $LUSCL$ [languages generated by local unordered scattered context grammars] (more on these equivalences can be found, e.g., in Engelfriet, 1997; Rambow and Satta, 1999; Weir, 1992).

In the present context the combination of algebraic, logical and regular techniques not only adds another characterization of mildly context-sensitive languages to the already long list of weak equivalences. It also makes available the whole body of techniques that have been developed in the tradition of algebraic language theory, logic and automata theory.

Extending the work by Michaelis (1999) which shows how to encode an arbitrary minimalist grammar (MG) in the sense of Stabler (1997) into a weakly equivalent linear context-free rewriting system (LCFRS), we present in this paper a translation from the formalism of multiple context-free grammars (MCFGs)—a weakly equivalent extension of LCFRSs—into regular tree grammars (RTGs). The idea behind the translation is to “lift” the MCFG rules to RTG rules by viewing them as Lawvere terms. Furthermore, we use the equivalence of RTGs, monadic second-order (MSO) logic (on trees) and tree automata to give an algebraic and a logical description of the lifted trees. Since the trees characterized by an RTG contain additional “non-linguistic” information they are then unpacked with a monadic second-order (MSO) transduction thereby giving both an operational and a denotational description of the tree sets involved. The MSO transduction is built upon a tree-walking automaton (with tests) at heart.

We think that our approach has decisive advantages. First, the operations of the relevant signature appear explicitly in the lifted trees and are not hidden in node labels coding instances of rule application. Second, our path component is not dependent on the particular regular tree family or, equivalently, the domain defined via the MSO formula. The instruction set of the tree-walking automaton and the corresponding definition of the MSO transduction are universal and only serve to reverse the lifting process. In that sense the instructions are nothing else but a restatement of the unique homomorphism which exists between the free algebra and any other algebra of the same signature. Thus, the translation from MCFGs to RTGs constitutes a considerable simplification in comparison with other characterizations since it is not built upon derivation trees using productions of the original MCFG as node labels, but rather on the operations of projection, tuple-formation and composition alone.

In the following sections we limit ourselves to the special case of MCFG rules with only one nonterminal on the right hand side (RHS). This allows a significant simplification in the presentation since it requires only one level of tupling. The extension to the general case of using tuples of tuples is considerably more involved and cannot be described here.

The structure of the paper is as follows. We start with some basic algebraic, logical and automata-theoretic definitions before turning to the formal presentation of MGs. The succeeding sections then sketch how to translate a given MG firstly into an MCFG and from there into an RTG. Finally, in the last section, we transform the resulting trees back into a format which is intended for linguistic analysis. We conclude with a brief outlook on further optimizations of the presented technique.

2 Background and Basic Definitions

We are going to show how to code the grammar rules of a LCFRS (or better an MCFG) into rules of an RTG. This is done via lifting by viewing MCFG rules as terms in a free Lawvere theory. Since this coding makes projection, tupling and composition explicit, the resulting trees contain these operations as labeled nodes. Therefore we use an MSO transduction—where the regular tree language constitutes the domain—to transform the lifted trees into the intended ones. Since the MSO formulas are implemented via automata, we also have to introduce the necessary types of automata.

In this section we present the corresponding basic algebraic, logical and automata-theoretic definitions before we proceed with the actual translation.

2.1 Basic Algebraic Definitions

Definition 2.1. For a given set of sorts $\mathcal{S}$,[1] a *many-sorted signature Σ (over $\mathcal{S}$)* is an indexed family $\langle \Sigma_{w,s} \mid w \in \mathcal{S}^*, s \in \mathcal{S} \rangle$ of disjoint sets. A symbol in $\Sigma_{w,s}$ is an *operator of type $\langle w, s \rangle$*, *arity w*, *sort s* and *rank $|w|$*.

The *set of trees $T(\Sigma)$ (over Σ)* is built up using the operators in the usual way: If $\sigma \in \Sigma_{\varepsilon,s}$ for some $s \in \mathcal{S}$ then σ is a (trivial) tree of sort s. If, for some $s \in \mathcal{S}$ and $w = s_1 \cdots s_n$ with $s_i \in \mathcal{S}$, $\sigma \in \Sigma_{w,s}$ and $t_1, \dots, t_n \in T(\Sigma)$ with t_i of sort s_i then $\sigma(t_1, \dots, t_n)$ is a tree of sort s.

In case $\mathcal{S}$ is a singleton $\{s\}$, i.e. in case Σ is a *single-sorted signature (over sort s)*, we usually write Σ_n to denote the (unique) set of operators of rank $n \in \mathbb{N}$.[2]

The operator symbols of a many-sorted signature Σ over some set of sorts $\mathcal{S}$ induce operations on an algebra with the appropriate structure. A *Σ-algebra* $\mathbb{A}$ consists of an $\mathcal{S}$-indexed family $\langle A^s \mid s \in \mathcal{S} \rangle$ of disjoint sets, the carriers of $\mathbb{A}$, and for each operator $\sigma \in \Sigma_{w,s}$, $\sigma_{\mathbb{A}} : A^w \to A^s$ is a function, where $A^w = A^{s_1} \times \cdots \times A^{s_n}$ and $w = s_1 \cdots s_n$ with $s_i \in \mathcal{S}$. The set $T(\Sigma)$ can be made into a Σ-algebra $\mathbb{T}$ by specifying the operations as follows. For every $\sigma \in \Sigma_{w,s}$, where $s \in \mathcal{S}$ and

[1] Throughout the paper the following conventions apply. $\mathbb{N}$ is the set of all non-negative integers. For any set M, M^* is the Kleene closure of M, i.e. the set of all finite strings over M. For $m \in M^*$, $|m| \in \mathbb{N}$ denotes the length of m. We will use M_ε to denote the set $M \cup \{\varepsilon\}$, where ε is the empty string (over M), i.e. $\varepsilon \in M^*$ with $|\varepsilon| = 0$.

[2] Note that for $\mathcal{S} = \{s\}$ each $\langle w, s \rangle \in \mathcal{S}^* \times \mathcal{S}$ is of the form $\langle s^n, s \rangle$ for some $n \in \mathbb{N}$.

$w = s_1 \cdots s_n$ with $s_i \in \mathcal{S}$, and every $t_1, \ldots, t_n \in T(\Sigma)$ with t_i of sort s_i we identify $\sigma_{\mathbb{T}}(t_1, \ldots, t_n)$ with $\sigma(t_1, \ldots, t_n)$. Our main notion is that of an algebraic (Lawvere) theory.

Definition 2.2. Given a set of sorts $\mathcal{S}$, an *algebraic (Lawvere) theory*, as an algebra, is an $\mathcal{S}^* \times \mathcal{S}^*$-sorted algebra $\mathbb{A}$ whose carriers $\langle A^{\langle u,v \rangle} \mid u, v \in \mathcal{S}^* \rangle$ consist of the morphisms of the theory and whose operations are of the following types, where $n \in \mathbb{N}$, $u = u_1 \cdots u_n$ with $u_i \in \mathcal{S}$ for $1 \leq i \leq n$ and $v, w \in \mathcal{S}^*$,

$$\begin{array}{ll} \text{projection:} & \pi_i^u \in A^{\langle u,u_i \rangle} \\ \text{composition:} & c_{(u,v,w)} \in A^{\langle u,v \rangle} \times A^{\langle v,w \rangle} \to A^{\langle u,w \rangle} \\ \text{target tupling:} & (\quad)_{(v,u)} \in A^{\langle v,u_1 \rangle} \times \cdots \times A^{\langle v,u_n \rangle} \to A^{\langle v,u \rangle} \end{array}$$

The projections and the operations of target tupling are required to satisfy the obvious identities for products. The composition operations must satisfy associativity, i.e.,

$$c_{(v,u,u_i)}\big((\alpha_1, \ldots, \alpha_n)_{(v,u)}, \pi_i^u\big) = \alpha_i \quad \text{for} \quad \alpha_i \in A^{\langle v,u_i \rangle}, 1 \leq i \leq n$$

$$\big(c_{(v,u,u_1)}(\beta, \pi_1^u), \ldots, c_{(v,u,u_n)}(\beta, \pi_n^u)\big)_{(v,u)} = \beta \quad \text{for} \quad \beta \in A^{\langle v,u \rangle}$$

$$c_{(u,v,z)}\big(\alpha, c_{(v,w,z)}(\beta, \gamma)\big) = c_{(u,w,z)}\big(c_{(u,v,w)}(\alpha, \beta), \gamma\big)$$
$$\text{for} \quad \alpha \in A^{\langle u,v \rangle}, \beta \in A^{\langle v,w \rangle}, \gamma \in A^{\langle w,z \rangle}$$

$$c_{(u,u,v)}\big((\pi_1^u, \ldots, \pi_n^u)_{(u,u)}, \alpha\big) = \alpha \quad \text{for} \quad \alpha \in A^{\langle u,v \rangle}$$

where $u = u_1 \cdots u_n$ with $u_i \in \mathcal{S}$ for $1 \leq i \leq n$ and $v, w, z \in \mathcal{S}^*$.

Let Σ be a single-sorted signature and $X = \{x_1, x_2, x_3, \ldots\}$ a countable set of variables. For $k \in \mathbb{N}$ define $X_k \subseteq X$ as $\{x_1, \ldots, x_k\}$. Then, the set of *k-ary trees* $T(\Sigma, X_k)$ *(over* Σ*)* is the set of trees $T(\Sigma')$ over the single-sorted signature $\Sigma' = \langle \Sigma'_n \mid n \in \mathbb{N} \rangle$, where $\Sigma'_0 = \Sigma_0 \cup X_k$ and $\Sigma'_n = \Sigma_n$ for $n > 0$. Note that $T(\Sigma, X_k) \subseteq T(\Sigma, X_l)$ for $k \leq l$. Let $T(\Sigma, X) = \bigcup_{k \in \mathbb{N}} T(\Sigma, X_k)$.

The power set $\wp(T(\Sigma,X))$ of $T(\Sigma, X)$ constitutes the central example of interest for formal language theory. The carriers $\langle \wp(T(k, m)) \mid k, m \in \mathbb{N} \rangle$ of the corresponding $\mathcal{S}^* \times \mathcal{S}^*$-Lawvere algebra are constituted by the power sets of the sets $T(k, m)$, where each $T(k, m)$ is the set of all m-tuples of k-ary trees, i.e. $T(k, m) = \{(t_1, \ldots, t_m) \mid t_i \in T(\Sigma, X_k)\}$.[3] For $i, k \in \mathbb{N}$ with $1 \leq i \leq k$ the projection constant π_i^k is defined as $\{x_i\}$. Composition is defined as substitution of the projection constants and target tupling is just tupling.

As remarked previously, an arbitrary number of nonterminals on the right hand side of an MCFG-rule entails the use of tuples of tuples in the definition of the corresponding mapping. That is to say, each nonterminal on the RHS generates a tuple of terminal strings rather than a single string (cf. Def. 4.1). Defining a corresponding Lawvere algebra for this general case requires to start with a many-sorted signature over $\mathbb{N}$ and a countable set of variables for each sort. Thus, the resulting Lawvere algebra is $\mathbb{N}^* \times \mathbb{N}^*$-sorted. Each carrier is of the form $\wp(T(\langle k_1, \ldots, k_l \rangle, \langle m_1, \ldots, m_n \rangle)$ with $k_i, m_j \in \mathbb{N}$, the power set of the set of n-tuples whose components are m_i-tuples of $\sum_{i=1}^{l} k_i$-ary trees. It seems obvious to us that this obfuscates the presentation to the point of unintelligibility.

More on Lawvere theories in general can be found in, e.g., Wagner (1994). More on the connection to linguistics is elaborated in Mönnich (1998).

2.2 Basic Logical Definitions

After these algebraic notions, we briefly present those related to monadic second-order (MSO) logic. MSO logic is the extension of first-order predicate logic with monadic second-order variables and quantification over them. In particular, we are using MSO logic on trees such that individual variables $x, y, \ldots$ stand for nodes in trees and monadic second-order ones $X, Y, \ldots$ for sets of nodes

[3]Since $\mathcal{S}$ is a singleton, $\mathcal{S}^*$ can be identified with $\mathbb{N}$, because up to length each $w \in \mathcal{S}^*$ is uniquely specified (cf. fn. 2).

(for more details see, e.g., Rogers, 1998). It is well-known that MSO logic interpreted on trees is decidable via a translation to finite-state (tree) automata (Rabin, 1969; Doner, 1970; Thatcher and Wright, 1968). The decidability proof for MSO on finite trees gives us also a descriptive complexity result: MSO on finite trees yields only recognizable trees which in turn yield context-free string languages. These results are of particular interest, since finite trees are clearly sufficient for linguistic purposes, and therefore form the basis for our work.

The following paragraphs go directly back to Courcelle (1997). Recall that the representation of objects within relational structures makes them available for the use of logical description languages. Let R be a finite set of relation symbols with the corresponding arity for each $r \in R$ given by $\rho(r)$. A relational structure $\mathcal{R} = \langle D_{\mathcal{R}}, (r_{\mathcal{R}})_{r \in \mathcal{R}} \rangle$ consists of the domain $D_{\mathcal{R}}$ and the $\rho(r)$-ary relations $r_{\mathcal{R}} \subseteq D_{\mathcal{R}}^{\rho(r)}$. In our case we choose a finite tree as our domain and the relations of immediate, proper and reflexive dominance and precedence.

There does not seem to be a convenient machine model for tree transformations. Fortunately, one can use logic directly to define the desired transduction. The classical technique of interpreting a relational structures within another one forms the basis for MSO transductions. Intuitively, the output tree is interpreted on the input tree. E.g., suppose that we want to transduce the input tree t_1 into the output tree t_2. The nodes of the output tree t_2 will be a subset of the nodes from t_1 specified with a unary MSO relation ranging over the nodes of t_1. The daughter relation will be specified with a binary MSO relation with free variables x and y ranging over the nodes from t_1. We will use this concept to transform the lifted trees into the intended ones.

Definition 2.3. A *(non-copying) MSO transduction* of a relational structure $\mathcal{R}$ (with set of relation symbols R) into another one $\mathcal{Q}$ (with set of relation symbols Q) is defined to be a tuple $(\varphi, \psi, (\theta_q)_{q \in Q})$. It consists of the formulas φ defining the domain of the transduction in $\mathcal{R}$ and ψ defining the resulting domain of $\mathcal{Q}$ and a family of formulas θ_q defining the new relations Q (using only definable formulas from the "old" structure $\mathcal{R}$).

The result which gives rise to the fact that we can characterize a non-context-free tree set with two devices which have only regular power is stated in Courcelle (1997). Viewing the relation of intended dominance defined later by a tree-walking automaton as the cornerstone of an MSO definable transduction, our description of non-context-free phenomena with two devices with only regular power is an instance of the theorem that the image of an MSO-definable class of structures under a definable transduction is not MSO definable in general (Courcelle, 1997).

2.3 Basic Automata-Theoretic Definitions

Tree automata are the result of generalizing the transition function of standard finite-state automata from (state-alphabet) symbol pairs to tuples of states. Intuitively, bottom-up tree automata creep up a tree from the leaves to the root by simultaneously taking the states of the daughters and the alphabet symbol of the mother to make a transition to a new state. Since we will not give an example of a tree automaton in this paper, we will not specify further details. Details on tree automata can be found in, e.g., Gécseg and Steinby (1984).

Definition 2.4 (Tree Automaton). A *(deterministic) bottom-up tree automaton* $\mathfrak{A}$ is a 5-tuple $\langle A, \Sigma, \delta, a_0, F \rangle$ with A the (finite) set of states, Σ a ranked alphabet, $a_0 \in A$ the initial state, $F \subseteq A$ the final states and $\delta : \bigcup_n (A^n \times \Sigma_n) \to A$ the transition function.

We can extend the transition function inductively to trees by defining $h_\delta(\varepsilon) = a_0$ and $h_\delta(\sigma(t_1, \dots, t_n)) = \delta(h_\delta(t_1), \dots, h_\delta(t_n), \sigma)$, $t_i \in T_\Sigma, 1 \leq i \leq n, \sigma \in \Sigma_n$. An automaton $\mathfrak{A}$ accepts a tree $t \in T_\Sigma$ iff $h_\delta(t) \in F$. The language recognized by $\mathfrak{A}$ is denoted by $T(\mathfrak{A}) = \{t \mid h_\delta(t) \in F\}$.

Furthermore, we have to introduce a concept which combines both automata theory and logic. We need a particular type of finite-state automaton: *tree-walking automata with MSO tests* (Bloem and Engelfriet, 1997). Intuitively, these automata make transitions from nodes in a tree to other nodes along its branches. Each transition can test a label of a node, move up in the tree or down to a specific daughter.

Definition 2.5 (Tree-Walking Automaton). A *tree-walking automaton (with tests)* over some many-sorted signature Σ is a finite automaton $\mathfrak{A} = (Q, \Delta, \delta, I, F)$ with states Q, directives Δ, transitions $\delta : Q \times \Delta \to Q$ and the initial and final states $I \subseteq Q$ and $F \subseteq Q$ which traverses a tree along connected edges using three kinds of directives: $\uparrow_i$—"move up to the mother of the current node (if it has one and it is its i-th daughter)", $\downarrow_i$—"move to the i-th daughter of the current node (if it exists)", and $\varphi(x)$—"verify that φ holds at the current node".

Let Σ be a many-sorted signature. For any tree $t \in T(\Sigma)$, a tree-walking automaton $\mathfrak{A}$ over Σ computes a node relation $R_t(\mathfrak{A}) = \{(x, y) | (x, q_i) \overset{*}{\Rightarrow} (y, q_f)$ for some $q_i \in I$ and some $q_f \in F\}$, where for all states $q_k, q_l \in Q$ and nodes x, y in t $(x, q_k) \Longrightarrow (y, q_l)$ iff $\exists d \in \Delta : (q_k, d, q_l) \in \delta$ and y is reachable from x in t via d. Note that x is reachable from itself if the directive was a (successful) test. It is important not to confuse this relation with the *walking language* recognized by the automaton, i.e., the string of directives needed to move from the initial to the final node in a walk. Bloem and Engelfriet show that these automata characterize the MSO definable node relations, i.e., every tree-walking automaton we specify can be inductively transformed into an equivalent MSO formula and vice versa.

3 Minimalist Grammars

We first give the definition of a minimalist grammar (MG) along the lines of Stabler (1997). In order to keep the representation simple, we omit the case of *strong selection* (triggering *head movement*), and *covert movement.* Thus the definition given here comes close to the one given in Stabler (1999), where some further restrictions are formulated as to which subtrees of a given tree may move. In fact, the example MG which will be considered below respects both the definition in Stabler (1997) as well as that in Stabler (1999).

Definition 3.1. For a given set (of features), *Feat*, a five-tuple $\tau = (N_\tau, \triangleleft^*_\tau, \prec_\tau, <_\tau, \mathit{Label}_\tau)$ fulfilling (E1)–(E3) is called an *expression (over Feat).*

(E1) $(N_\tau, \triangleleft^*_\tau, \prec_\tau)$ is a finite, binary ordered tree. N_τ denotes the non-empty set of nodes. $\triangleleft^*_\tau$ and $\prec_\tau$ denote the usual relations of *dominance* and *precedence* defined on a subset of $N_\tau \times N_\tau$, respectively. I.e. $\triangleleft^*_\tau$ is the reflexive and transitive closure of $\triangleleft_\tau$, the relation of *immediate dominance.*[4]

(E2) $<_\tau \subseteq N_\tau \times N_\tau$ denotes the asymmetric relation of *(immediate) projection* which holds for any two siblings in $(N_\tau, \triangleleft^*_\tau, \prec_\tau)$, i.e. each node different from the root either *(immediately) projects* over its sibling or vice versa.

(E3) The function Label_τ assigns a string from Feat^* to every leaf of $(N_\tau, \triangleleft^*_\tau, \prec_\tau)$, i.e. a leaf-label is a finite sequence of features from *Feat.*

The set of all expressions over *Feat* is denoted by *Exp*(*Feat*).

Let *Feat* be a set of features. Consider $\tau = (N_\tau, \triangleleft^*_\tau, \prec_\tau, <_\tau, \mathit{Label}_\tau) \in \mathit{Exp}(\mathit{Feat})$.

Each $x \in N_\tau$ has a *head* $h(x) \in N_\tau$, a leaf such that $x \triangleleft^*_\tau h(x)$, and such that each $y \in N_\tau$ on the path from x to $h(x)$ with $y \neq x$ projects over its sister. The *head of* τ is the head of τ's root.

A subtree υ of τ is a *maximal projection (in* τ*)*, if the root of υ is a node $x \in N_\tau$ such that x is the root of τ or x's sister projects over x. The sister of the head of τ is the *complement (of* τ*).* Each maximal projection in τ which is not dominated by the mother of the head of τ is a *specifier (of* τ*).*

τ *has feature* $f \in \mathit{Feat}$ if τ's head-label starts with f. τ is *simple* (a *head*) if it consists of exactly one node, otherwise τ is *complex* (a *non-head*).

[4] Up to an isomorphism N_τ is a unique *prefix closed* and *left closed* subset of $\mathbb{N}^*$, i.e. $\chi \in N_\tau$ if $\chi\chi' \in N_\tau$, and $\chi i \in N_\tau$ if $\chi j \in N_\tau$ for $\chi, \chi' \in \mathbb{N}^*$ and $i, j \in \mathbb{N}$ with $i < j$, such that for $\chi, \psi \in N_\tau$ hold: $\chi \triangleleft_\tau \psi$ iff $\psi = \chi i$ for some $i \in \mathbb{N}$, and $\chi \prec_\tau \psi$ iff $\chi = \omega i \chi'$ and $\psi = \omega j \psi'$ for some $\omega, \chi', \psi' \in \mathbb{N}^*$ and $i, j \in \mathbb{N}$ with $i < j$.

Let r_τ be the root of τ. Suppose υ and $\varphi \in \mathit{Exp}(\mathit{Feat})$ to be subtrees of τ with roots r_υ and r_φ, respectively, such that $r_\tau \triangleleft_\tau r_\upsilon, r_\varphi$. Then we take $[_<\upsilon, \varphi\,]$ $(\,[_>\varphi, \upsilon\,]\,)$ to denote τ in case that $r_\upsilon <_\tau r_\varphi$ and $r_\upsilon \prec_\tau r_\varphi$ $(r_\varphi \prec_\tau r_\upsilon)$.

Definition 3.2 (Stabler, 1997). A 4-tuple
$G = \langle \mathit{Non\text{-}Syn}, \mathit{Syn}, \mathit{Lex}, \mathcal{F}\rangle$ that obeys (M1)–(M4) is called a *minimalist grammar (MG).*

(M1) *Non-Syn* is a finite set of *non-syntactic features* partitioned into a set *Phon* of *phonetic features* and a set *Sem* of *semantic features.*

(M2) *Syn* is a finite set of *syntactic features* partitioned into the sets *Base*, *Select*, *Licensees* and *Licensors* such that for each *(basic) category* $\mathtt{x} \in \mathit{Base}$ the existence of ${}^{=}\mathtt{x} \in \mathit{Select}$ is possible, and for each $\mathtt{-x} \in \mathit{Licensees}$ the existence of $\mathtt{+X} \in \mathit{Licensors}$ is possible. Moreover, the set *Base* contains at least the category $\mathtt{c}$.

(M3) *Lex* is a finite set of expressions over $\mathit{Feat} = \mathit{Non\text{-}Syn} \cup \mathit{Syn}$ such that for each tree $\tau = \langle N_\tau, \triangleleft^*_\tau, \prec_\tau, <_\tau, \mathit{Label}_\tau\rangle \in \mathit{Lex}$ the function Label_τ assigns a string from $\mathit{Select}^*\,\mathit{Licensors}_\varepsilon\,\mathit{Select}^*\,\mathit{Base}_\varepsilon\,\mathit{Licensees}^*\,\mathit{Phon}^*\,\mathit{Sem}^* \subseteq \mathit{Feat}^*$ to each leaf in $\langle N_\tau, \triangleleft^*_\tau, \prec_\tau\rangle$.

(M4) The set $\mathcal{F}$ consists of the structure building functions *merge* and *move* as defined in (me) and (mo), respectively.

(me) The function *merge* is a partial mapping from $\mathit{Exp}(\mathit{Feat}) \times \mathit{Exp}(\mathit{Feat})$ to $\mathit{Exp}(\mathit{Feat})$. A pair of expressions (υ, φ) belongs to $\mathrm{Dom}(\mathit{merge})$ if υ has feature ${}^{=}\mathtt{x}$ and φ has category $\mathtt{x}$ for some $\mathtt{x} \in \mathit{Base}$.[5] Then,

(me.1) $$\mathit{merge}(\upsilon, \varphi) = \ [_<\upsilon', \varphi'\,] \quad \text{if } \upsilon \text{ is simple and has feature } {}^{=}\mathtt{x},$$

where υ' and φ' are expressions resulting from υ and φ, respectively, by deleting the feature the respective head-label starts with.

(me.2) $$\mathit{merge}(\upsilon, \varphi) = \ [_>\varphi', \upsilon'\,] \quad \text{if } \upsilon \text{ is complex and has feature } {}^{=}\mathtt{x},$$

where υ' and φ' are expressions as in case (me.1).

(mo) The function *move* is a partially defined mapping from $\mathit{Exp}(\mathit{Feat})$ to $\mathit{Exp}(\mathit{Feat})$. An expression υ belongs to $\mathrm{Dom}(\mathit{move})$ in case that υ has feature $\mathtt{+X} \in \mathit{Licensors}$, and υ has exactly one subtree φ that is a maximal projection and has feature $\mathtt{-x} \in \mathit{Licensees}$. Then,

$$\mathit{move}(\upsilon) = \ [_>\varphi', \upsilon'\,] \quad \text{if } \upsilon \text{ has feature } \mathtt{+X}$$

Here υ' results from υ by deleting the feature $\mathtt{+X}$ from υ's head-label, while the subtree φ is replaced by a single node labeled ε. φ' is the expression resulting from φ just by deleting the licensee feature $\mathtt{-x}$ that φ's head-label starts with.

Note that, by (me.1) and (me.2), a simple tree (head) selects another tree as its complement to the right, whereas a complex tree selects another tree as a specifier to the left.

Example 3.3. Let G_{ww} be the MG with $\mathit{Sem} = \emptyset$, $\mathit{Phon} = \{1, 2\}$, $\mathit{base} = \{\mathtt{c}, \mathtt{a}_1, \mathtt{a}_2, \mathtt{b}, \mathtt{c}_1, \mathtt{c}_2, \mathtt{d}\}$, $\mathit{select} = \{{}^{=}\mathtt{a}_1, {}^{=}\mathtt{a}_2, {}^{=}\mathtt{b}, {}^{=}\mathtt{c}_1, {}^{=}\mathtt{c}_2, {}^{=}\mathtt{d}\}$, $\mathit{Licensors} = \{\mathtt{+L}_1, \mathtt{+L}_2\}$, $\mathit{Licensees} = \{\mathtt{-l}_1, \mathtt{-l}_2\}$, while *Lex* consists of the following 10 simple expressions, where $i \in \{1, 2\}$,[6]

$$\begin{array}{lll} \alpha_i = \mathtt{a}_i\mathtt{-l}_1 i & \gamma_i = {}^{=}\mathtt{b}\mathtt{+L}_1\mathtt{c}_i\mathtt{-l}_1 i & \zeta_1 = {}^{=}\mathtt{b}\mathtt{+L}_1\mathtt{d} \\ \beta_i = {}^{=}\mathtt{a}_i\mathtt{b}\mathtt{-l}_2 i & \delta_i = {}^{=}\mathtt{c}_i\mathtt{+L}_2\mathtt{b}\mathtt{-l}_2 i & \zeta_2 = {}^{=}\mathtt{d}\mathtt{+L}_2\mathtt{c} \end{array}$$

Then e.g., for $i, j \in \{1, 2\}$, $\mathit{move}(\mathit{merge}(\delta_j, \mathit{move}(\mathit{merge}(\gamma_j, \mathit{merge}(\beta_i, \alpha_i))))) \in \mathit{Exp}(\mathit{Feat})$.

[5]For each (partial) mapping f from a set M_1 into a set M_2 we take $\mathrm{Dom}(f)$ to denote the *domain of* f, the subset of M_1 for which f is defined.

[6]Since all lexical entries are heads, we simply represent them by their respective (unique) labels.

Let $G = (\mathit{Non\text{-}Syn}, \mathit{Syn}, \mathit{Lex}, \mathcal{F})$ be an MG. Then $CL(G) = \bigcup_{k \in \mathbb{N}} CL^k(G)$ is the *closure of Lex (under the functions in $\mathcal{F}$)*. For $k \in \mathbb{N}$ the sets $CL^k(G) \subseteq \mathit{Exp}(\mathit{Feat})$ are inductively defined by

$$\begin{aligned}
CL^0(G) &= \mathit{Lex} \\
CL^{k+1}(G) &= CL^k(G) \\
&\quad \cup \{\mathit{merge}(\upsilon, \varphi) \mid (\upsilon, \varphi) \in \mathrm{Dom}(\mathit{merge}) \cap CL^k(G) \times CL^k(G)\} \\
&\quad \cup \{\mathit{move}(\upsilon) \mid \upsilon \in \mathrm{Dom}(\mathit{move}) \cap CL^k(G)\}
\end{aligned}$$

Every $\tau \in CL(G)$ is called an *expression in G*. Such a τ is *complete (in G)* if its head-label is in $\{\mathtt{c}\}\mathit{Phon}^*\mathit{Sem}^*$ and each other of its leaf-labels is in $\mathit{Phon}^*\mathit{Sem}^*$. Hence, a complete expression has category $\mathtt{c}$, and this instance of $\mathtt{c}$ is the only instance of a syntactic feature within all leaf-labels.

The *(phonetic) yield* $Y(\tau)$ of an expression $\tau \in \mathit{Exp}(\mathit{Feat})$ is the string created by concatenating τ's leaf-labels "from left to right" and stripping off all non-phonetic features. $L(G) = \{Y(\tau) \mid \tau \in CL(G)$ with τ is complete$\}$ is the *(string) language (derivable by G)* and is called a *minimalist language.*

Example 3.3 (continued) For $i \in \{1,2\}$ and $u \in \{1,2\}^+$ the expressions belonging to $CL(G_{ww})$ can recursively be defined by

(1a) $\omega_i = \mathit{merge}(\beta_i, \alpha_i)$

(1b) $\varphi_{iu} = \mathit{merge}(\gamma_i, \omega_u)$ (1b') $\eta_u = \mathit{merge}(\zeta_1, \omega_u)$

(1c) $\chi_{iu} = \mathit{move}(\varphi_{iu})$ (1c') $\vartheta_u = \mathit{move}(\eta_u)$

(1d) $\psi_{iu} = \mathit{merge}(\delta_i, \chi_{iu})$ (1d') $\kappa_u = \mathit{merge}(\zeta_2, \vartheta_u)$

(1e) $\omega_{iu} = \mathit{move}(\psi_{iu})$ (1e') $\xi_u = \mathit{move}(\kappa_u)$

In more detail, we have

(2a) $CL^1(G_{ww}) \setminus CL^0(G_{ww}) = \{\omega_1, \omega_2\}$

while for $k \in \mathbb{N}$ we have

(2b) $CL^{4k+2}(G_{ww}) \setminus CL^{4k+1}(G_{ww}) = \{\varphi_{iu}, \eta_u \mid i \in \{1,2\}, u \in \{1,2\}^+ \text{ with } |u| = k\}$

(2c) $CL^{4k+3}(G_{ww}) \setminus CL^{4k+2}(G_{ww}) = \{\chi_{iu}, \vartheta_u \mid i \in \{1,2\}, u \in \{1,2\}^+ \text{ with } |u| = k\}$

(2d) $CL^{4k+4}(G_{ww}) \setminus CL^{4k+3}(G_{ww}) = \{\psi_{iu}, \kappa_u \mid i \in \{1,2\}, u \in \{1,2\}^+ \text{ with } |u| = k\}$

(2e) $CL^{4k+5}(G_{ww}) \setminus CL^{4k+4}(G_{ww}) = \{\omega_{iu}, \xi_u \mid i \in \{1,2\}, u \in \{1,2\}^+ \text{ with } |u| = k\}$

The set of complete expressions in G_{ww} is $\{\xi_u \mid u \in \{1,2\}^+\}$. Each such ξ_u has the phonetic yield $Y(\xi_u) = uu$, i.e. the string language derivable by G_{ww} is $\{uu \mid u \in \{1,2\}^+\}$.

Definition 3.4. For each MG $G = \langle \mathit{Non\text{-}Syn}, \mathit{Syn}, \mathit{Lex}, \mathcal{F} \rangle$, an expression $\tau \in CL(G)$ is called *relevant (in G)* if it has property (R).

(R) For any $\mathtt{-x} \in \mathit{Licensees}$ there is at most one maximal projection $\tau_{\mathtt{-x}}$ in τ that has feature $\mathtt{-x}$.[7]

For any given MG G, we take $\mathit{Rel}(G)$ the set of all relevant expressions $\tau \in CL(G)$. Since each complete $\tau \in CL(G)$ has property (R), we have $L(G) = \{Y(\tau) \mid \tau \in \mathit{Rel}(G),\ \tau \text{ is complete}\}$.

Remark 3.5. For G_{ww} as in Example 3.3 we have $\mathit{Rel}(G_{ww}) = CL(G_{ww})$.

[7] In fact, this kind of structure is characteristic of each $\tau \in CL(G)$ involved in creating a complete expression in G. Recall that $\mathit{move}(\tau)$ is defined for $\tau \in CL(G)$ only in case that there is exactly one maximal subtree of τ that has a particular licensee feature allowing the subtree's "movement into specifier position."

4 Translating MGs to MCFGs

In Michaelis (1999) an algorithm is given how to transform an arbitrary MG G into a weakly equivalent MCFG. The core idea is that for $Rel(G)$, i.e. the set of trees appearing as intermediate steps in converging derivations of G, one can define a finite partition. The equivalence classes of this partition are formed by sets of trees where the features triggering movement appear in identical structural positions. Each nonterminal in a corresponding MCFG represents such an equivalence class, i.e., an infinite set of trees. In this paper we will concentrate on the example from above, adopting the methods from Michaelis (1999).

Definition 4.1. A *multiple context-free grammar* (MCFG) is defined as a five-tuple $\mathcal{G} = \langle V_N, V_T, V_F, P, S\rangle$ with V_N, V_T, V_F and P being a finite set of ranked nonterminals, terminals, linear basic morphisms and productions, respectively. $S \in V_N$ is the start symbol, which has rank 1. Each $p \in P$ has the form $A \longrightarrow f(A_0, \dots, A_{n-1})$ for $A, A_0, \dots, A_{n-1} \in V_N$ and $f \in F$ a function from $(V_T^*)^k$ to $(V_T^*)^{d(A)}$ with arity $k = \sum_{i=0}^{n-1} d(A_i)$ ($d(A_i)$ the rank of A_i) and $d(A)$ the rank of A (cf. Seki et al. 1991). Recall that basic morphisms are those which use only variables, constants, concatenation, composition and tupling.

The derive-relation $\Rightarrow_\mathcal{G}$ for $\mathcal{G}$ is defined as follows: If $A \longrightarrow f() \in P$ then $A \Rightarrow_\mathcal{G} f()$, where $f() \in (V_T^*)^{d(A)}$ (i.e. f is some constant tuple of terminal strings). If $A \longrightarrow f(A_0, \dots, A_{n-1}) \in P$ and $A_i \Rightarrow_\mathcal{G} t_i$ for some $t_i \in (V_T^*)^{d(A_i)}$ then $A_i \Rightarrow_\mathcal{G} f(t_0, \dots, t_{n-1})$. The language generated by $\mathcal{G}$ is $L(\mathcal{G}) = \{t \in V_T^* \mid S \Rightarrow_\mathcal{G} t\}$.

Example 3.3 (continued) Let G_{ww} be the MG as given in Example 3.3. In order to define a weakly equivalent MCFG $\mathcal{G}_{ww} = \langle V_N, V_T, V_F, P, S\rangle$, we first let $Phon = \{1, 2\}$ be the set of terminals V_T and proceed by constructing the set of nonterminals V_N.

Each nonterminal will either be the start symbol S or a 3-tuple from $Syn^* \times Syn^* \times Syn^*$. The leading idea is the following: Consider $\tau \in CL(G_{ww}) \setminus CL^0(G_{ww})$. For $1 \leq i \leq 2$ take, if it exists, τ_i to be a subtree of τ that is a maximal projection and has licensee $\texttt{-l}_i$.[8] Otherwise, take τ_i to be a single node labeled ε. Set $\tau_0 = \tau$. Let μ_i be the prefix of τ_i's head-label consisting of just the syntactic features. Then, $\langle \mu_0, \mu_1, \mu_2\rangle \in V_N$. The productions (and functions) of the MCFG $\mathcal{G}_{ww}$ will be defined in such a way that for each $\langle p_0, p_1, p_2\rangle \in Phon^* \times Phon^* \times Phon^*$, $\langle \mu_0, \mu_1, \mu_2\rangle \Rightarrow_{\mathcal{G}_{ww}} \langle p_0, p_1, p_2\rangle$ iff (wc) holds.

(wc) For $0 \leq i \leq 2$, p_i is the phonetic yield of τ_i except for each substring that is the phonetic yield of some τ_j, $1 \leq j \leq 3$ and $i \neq j$, being a proper subtree of τ_i.

Although $CL(G_{ww}) \setminus CL^0(G_{ww})$ is an infinite set, V_N is only finite. This is due to two reasons emerging from the definition of *merge* and *move*. First, in each $\tau \in CL(G_{ww})$ at most 3 different leaves are present which have syntactic features appearing in their labels.[9] Second, for each MG G the set of all leaf-labels of all $\tau \in CL(G)$ constitutes a finite set, because a leaf-label is always the suffix of some leaf-label of some lexical entry, i.e. a suffix of a finite string, and the lexicon of an MG is a finite subset of $Exp(Non\text{-}Syn \cup Syn)$. In fact, we have exactly 10 nonterminals different from the start symbol S, each of which corresponds to an infinite set of expressions from $CL(G_{ww})$, namely[10]

(3a)	$U = \langle \texttt{b-l}_2, \texttt{-l}_1, - \rangle$	to	$\{\omega_i \mid i \in \{1,2\}\}$
(3b)	$V_i = \langle \texttt{+L}_1\texttt{c}_i\texttt{-l}_1, \texttt{-l}_1, \texttt{-l}_2 \rangle$	to	$\{\varphi_{iu} \mid u \in \{1,2\}^+\}$, where $i \in \{1,2\}$
(3c)	$U_i = \langle \texttt{c}_i\texttt{-l}_1, -, \texttt{-l}_2 \rangle$	to	$\{\chi_{iu} \mid u \in \{1,2\}^+\}$, where $i \in \{1,2\}$

[8]Recall that $Rel(G_{ww}) = CL(G_{ww})$. Therefore, such a τ_i is unique. In order to ensure this in general, we would have to reduce the closure of an MG G to what is defined as the *relevant closure* $RCL(G)$ of G in Michaelis (1999).

[9]Recall again Remark 3.5.

[10]Here, we write $-$ instead of ε.

(3d) $V = \langle \texttt{+L}_2\texttt{b-l}_2, \texttt{-l}_1, \texttt{-l}_2 \rangle$ to $\{\psi_{iu} \,|\, u \in \{1,2\}^+, i \in \{1,2\}\}$

(3e) $U = \langle \texttt{b-l}_2, \texttt{-l}_1, - \rangle$ to $\{\omega_{iu} \,|\, u \in \{1,2\}^+, i \in \{1,2\}\}$

(3b') $W_1 = \langle \texttt{+L}_1\texttt{d}, \texttt{-l}_1, \texttt{-l}_2 \rangle$ to $\{\eta_u \,|\, u \in \{1,2\}^+\}$

(3c') $X_1 = \langle \texttt{d}, -, \texttt{-l}_2 \rangle$ to $\{\vartheta_u \,|\, u \in \{1,2\}^+\}$

(3d') $W_2 = \langle \texttt{+L}_2\texttt{c}, -, \texttt{-l}_2 \rangle$ to $\{\kappa_u \,|\, u \in \{1,2\}^+\}$

(3e') $C = \langle \texttt{c}, -, - \rangle$ to $\{\xi_u \,|\, u \in \{1,2\}^+\}$

Thus, the nonterminals different from S introduce a finite partition of $CL(G_{ww}) \setminus CL^0(G_{ww})$. We now define P and F, the sets of productions and functions in $\mathcal{G}_{ww}$, respectively. Each $p \in P$ in a certain way simulates an application of *merge* and *move* in G_{ww}, but operates w.r.t. the equivalence classes of the induced partition, rather than on single expressions. As indicated in the introduction, we present in this paper only the significantly simpler case of using only MCFG rules with one nonterminal on the RHS. Although *merge* in principle is a binary operation, i.e., would require two nonterminals, we can in this special case partially evaluate the merge operation. This becomes possible since we only have "simple merges," i.e, we merge always a head with some complex tree, as opposed to merging two complex trees.

Let $i \in \{1,2\}$. P consists of 2 terminating rules,

(4a) $U \to \mathit{merge}_{\langle \beta_i, \alpha_i \rangle}()$

and 12 nonterminating rules,

(4b) $V_i \to \mathit{merge}_{\langle \gamma_i, \omega_{\_} \rangle}(U)$ (4b') $W_1 \to \mathit{merge}_{\langle \zeta_1, \omega_{\_} \rangle}(U)$

(4c) $U_i \to \mathit{move}_{\langle \varphi_{\_} \rangle}(V_i)$ (4c') $X_1 \to \mathit{move}_{\langle \eta_{\_} \rangle}(W_1)$

(4d) $V \to \mathit{merge}_{\langle \delta_i, \chi_{\_} \rangle}(U_i)$ (4d') $W_2 \to \mathit{merge}_{\langle \zeta_2, \vartheta_{\_} \rangle}(X_1)$

(4e) $U \to \mathit{move}_{\langle \psi_{\_} \rangle}(V)$ (4e') $C \to \mathit{move}_{\langle \kappa_{\_} \rangle}(W_2)$

(4f') $S \to \pi_1^3(C)$ (the initial rule)

The 9 basic functions in F are defined as follows:

(5a) $\mathit{merge}_{\langle \beta_i, \alpha_i \rangle} : (\{1,2\}^*)^0 \to (\{1,2\}^*)^3$ with $\langle\,\rangle \mapsto \langle i, i, \epsilon \rangle$ for $i \in \{1,2\}$

(5b) $\mathit{merge}_{\langle \gamma_i, \omega_{\_} \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle i, x_2, x_1 \rangle$ for $i \in \{1,2\}$

(5c) $\mathit{move}_{\varphi_{\_}} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle x_2 x_1, \epsilon, x_3 \rangle$

(5d) $\mathit{merge}_{\langle \delta_i, \chi_{\_} \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle i, x_1, x_3 \rangle$ for $i \in \{1,2\}$

(5e) $\mathit{move}_{\psi_{\_}} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle x_3 x_1, x_2, \epsilon \rangle$

(5b') $merge_{\langle \zeta_1, \omega_- \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle \epsilon, x_2, x_1 \rangle$

(5c') $move_{\eta_-} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle x_2 x_1, \epsilon, x_3 \rangle$

(5d') $merge_{\langle \zeta_2, \vartheta_- \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle x_1, \epsilon, x_3 \rangle$

(5e') $move_{\kappa_-} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3$ with $\langle x_1, x_2, x_3 \rangle \mapsto \langle x_3 x_1, x_2, \epsilon \rangle$

(5f') $\pi^3_1 : (\{1,2\}^*)^3 \to \{1,2\}^*$ with $\langle x_1, x_2, x_3 \rangle \mapsto x_1$

Note that $move_{\varphi_-} = move_{\eta_-}$ and $move_{\psi_-} = move_{\kappa_-}$. The 3-tuples of terminal strings derivable from the nonterminals different from S are the following, where $i \in \{1,2\}$ and $u \in \{1,2\}^+$,

(6a) $U \Rightarrow_{\mathcal{G}_{ww}} \langle i, i, \epsilon \rangle$

(6b) $V_i \Rightarrow_{\mathcal{G}_{ww}} \langle i, u, u \rangle$ (6b') $W_1 \Rightarrow_{\mathcal{G}_{ww}} \langle \epsilon, u, u \rangle$

(6c) $U_i \Rightarrow_{\mathcal{G}_{ww}} \langle ui, \epsilon, u \rangle$ (6c') $X_1 \Rightarrow_{\mathcal{G}_{ww}} \langle u, \epsilon, u \rangle$

(6d) $V \Rightarrow_{\mathcal{G}_{ww}} \langle i, ui, u \rangle$ (6d') $W_2 \Rightarrow_{\mathcal{G}_{ww}} \langle u, \epsilon, u \rangle$

(6e) $U \Rightarrow_{\mathcal{G}_{ww}} \langle iu, iu, \epsilon \rangle$ (6e') $C \Rightarrow_{\mathcal{G}_{ww}} \langle uu, \epsilon, \epsilon \rangle$

Thus, we finally have: (6f') $S \Rightarrow_{\mathcal{G}_{ww}} uu$ iff $u \in \{1,2\}^+$.

5 Translating MCFGs to RTGs

In this section we will show how to translate the rules of a given MCFG into an RTG. We start by giving a formal definition of regular tree grammars.

Definition 5.1. A *regular tree grammar* (RTG) is a 4-tuple $\mathcal{G} = \langle \Sigma, \mathsf{F}_0, S, \mathsf{P} \rangle$, where for some set of sorts $\mathcal{S}$, $\Sigma = \langle \Sigma_{w,s} \,|\, w \in \mathcal{S}^*, s \in \mathcal{S} \rangle$ is a many-sorted signature of *inoperatives* and $\mathsf{F}_0 = \langle \mathsf{F}_{\varepsilon,s} \,|\, s \in \mathcal{S} \rangle$ a (reduced) many-sorted signature of *operatives* of rank 0. Moreover, $\bigcup_{(w,s) \in \mathcal{S}^* \times \mathcal{S}} \Sigma_{w,s}$ and $\bigcup_{s \in \mathcal{S}} \mathcal{F}_{\varepsilon,s}$ are finite. $S \in \mathsf{F}_0$ is the starting symbol and P is a finite set of productions. Each $p \in \mathsf{P}$ has the form $F \longrightarrow t$, where $F \in \mathsf{F}_{\varepsilon,s}$ for some $s \in \mathcal{S}$ and $t \in T(\Sigma \cup \mathsf{F}_0)$, i.e. a term (tree) over $\Sigma \cup \mathsf{F}_0$, such that t is of sort s.

Let $t', t'' \in T(\Sigma \cup \mathsf{F}_0)$ and $p = F \longrightarrow t \in P$. t' *directly derives* t'' *(by the application of* p*)*, also denoted by $t' \Rightarrow_{\mathcal{G}} t''$, if t' has a leaf-node F and t'' results from t' by substituting this node F by t. Let $\Rightarrow^*_{\mathcal{G}}$ be the reflexive and transitive closure of $\Rightarrow_{\mathcal{G}}$. The tree-language generated by $\mathcal{G}$ is the set $L_T(\mathcal{G}) = \{t \in T(\Sigma) \,|\, S \Rightarrow^*_{\mathcal{G}} t\}$.

The yield $Y(t)$ of a $t \in T(\Sigma \cup \mathsf{F}_0)$ is the string resulting from concatenating the leaf-nodes of t "from left to right." Thus, $Y(t) \in (\bigcup_{s \in \mathcal{S}} \Sigma_{\varepsilon,s} \cup \bigcup_{s \in \mathcal{S}} \mathcal{F}_{\varepsilon,s})^*$. The string-language generated by $\mathcal{G}$ is the set $L_Y = \{Y(t) \,|\, t \in L_T(\mathcal{G})\} \subseteq (\bigcup_{s \in \mathcal{S}} \Sigma_{\varepsilon,s})^*$.

Since RTG rules always just substitute some tree for a leaf-node, it is easy to see that they generate recognizable sets of trees, i.e., context-free string languages (Mezei and Wright, 1967).

Now we turn to the actual translation. Each rule of a given MCFG is recursively transformed into a RTG rule by coding the implicit operations of projection, tupling and composition as nonterminals or terminals. This becomes possible simply by viewing the terms appearing in the rules of the MCFG as elements of a free $\mathbb{N} \times \mathbb{N}$-sorted Lawvere algebra. The resulting RTG then "operates on" this Lawvere algebra.

Example 3.3 (continued) As an example, we show how to translate the MCFG $\mathcal{G}_{ww}$ given in Ex. 3.3 into the weakly equivalent RTG. Intuitively, we have to make the implicit operations which are "hidden" in the standard presentation of the MCFG rules explicit. Simply using a tuple, e.g., the pair $\langle a, b\rangle$, means that we need an explicit tupling operator $(\ )$ to combine a and b. In the same spirit, using x_0x_1 means that the values of the two variables are concatenated with an implicit concatenation operator. And finally, applying a function to some arguments is a composition $\circ$ of the function with its arguments. Note that thereby the function becomes a constant. In this sense, a term such as $f(a, b)$ becomes more complex: $(f \circ ((\ )(x_1, x_2))) \circ (\ )(a, b)$. Now we have to translate these single-sorted terms into the corresponding many-sorted Lawvere algebra.

For $1 \leq i \leq 3$, let π^3_i denote the i-th projection which maps a 3-tuple of strings from V_T^* to its i-th component, i.e. a 1-tuple. Therefore the corresponding Lawvere arity of π^3_1, π^3_2 and π^3_3 is $(3, 1)$. Let $\bullet$ denote the usual binary operation of concatenation defined for strings from V_T^*, i.e., $\bullet$ maps a 2-tuple to a 1-tuple. Thus $\bullet$ is of Lawvere arity $(2, 1)$. Similarly, the corresponding (Lawvere) arity of $S, 1, 2$ and ε is $(0, 1)$ and of $U, V, C, W_1, W_2, X_1, U_i, V_i$ $(0, 3)$.

We use the composition symbols $c_{(i,j,k)}$ introduced in Sec. 2.1 for the composition $\circ$.

Let $\mathcal{G}_{ww} = \langle V_N, V_T, V_F, P, S\rangle$ be the MCFG successively constructed in the preceeding section. Applying the translation $\mathsf{T} : P \longrightarrow \mathsf{F}_0 \times T(\Sigma \cup \mathsf{F}_0)$ given below to the rules of the MCFG $\mathcal{G}_{ww}$ results in the RTG $\mathcal{G}'_{ww} = \langle \Sigma, \mathsf{F}_0, S_{(0,1)}, \mathsf{P}\rangle$ with inoperatives $\Sigma = \langle \Sigma_{w,s} \mid w \in (\mathbb{N}\times\mathbb{N})^*, s \in \mathbb{N}\times\mathbb{N}\rangle$, operatives F_0 of rank 0, and productions P which (in tree notation) look as given in Fig. 1, where $i \in \{1, 2\}$.[11] We have $\Sigma_{\varepsilon,(3,0)} = \{(\ )_{(3,0)}\}$, $\Sigma_{\varepsilon,(2,1)} = \{\bullet_{(2,1)}\}$, $\Sigma_{\varepsilon,(0,1)} = \{1_{(0,1)}, 2_{(0,1)}, \varepsilon_{(0,1)}\}$, $\Sigma_{\varepsilon,(3,1)} = \{\pi^3_{1\,(3,1)}, \pi^3_{2\,(3,1)}, \pi^3_{3\,(3,1)}\}$,

$$\begin{array}{lcl@{\qquad}lcl}
\Sigma_{(0,3)(3,3),(0,3)} & = & \{c_{(0,3,3)}\} & \Sigma_{(3,1)(3,1),(3,2)} & = & \{(\ )_{(3,2)}\} \\
\Sigma_{(0,3)(3,1),(0,1)} & = & \{c_{(0,3,1)}\} & \Sigma_{(0,1)(0,1)(0,1),(0,3)} & = & \{(\ )_{(0,3)}\} \\
\Sigma_{(3,2)(2,1),(3,1)} & = & \{c_{(3,2,1)}\} & \Sigma_{(3,1)(3,1)(3,1),(3,3)} & = & \{(\ )_{(3,3)}\} \\
\Sigma_{(3,0)(0,1),(3,1)} & = & \{c_{(3,0,1)}\} & & &
\end{array}$$

and $\mathsf{F}_0 = F_{\varepsilon,(0,1)} \cup F_{\varepsilon,(0,3)}$ with $F_{\varepsilon,(0,1)} = \{S_{(0,1)}\}$ and $F_{\varepsilon,(0,3)} = \{U_{(0,3)}, U_{i(0,3)}, V_{(0,3)}, V_{i(0,3)}, W_{1(0,3)}, W_{2(0,3)}, X_{1(0,3)}, C_{(0,3)}\}$.[12]

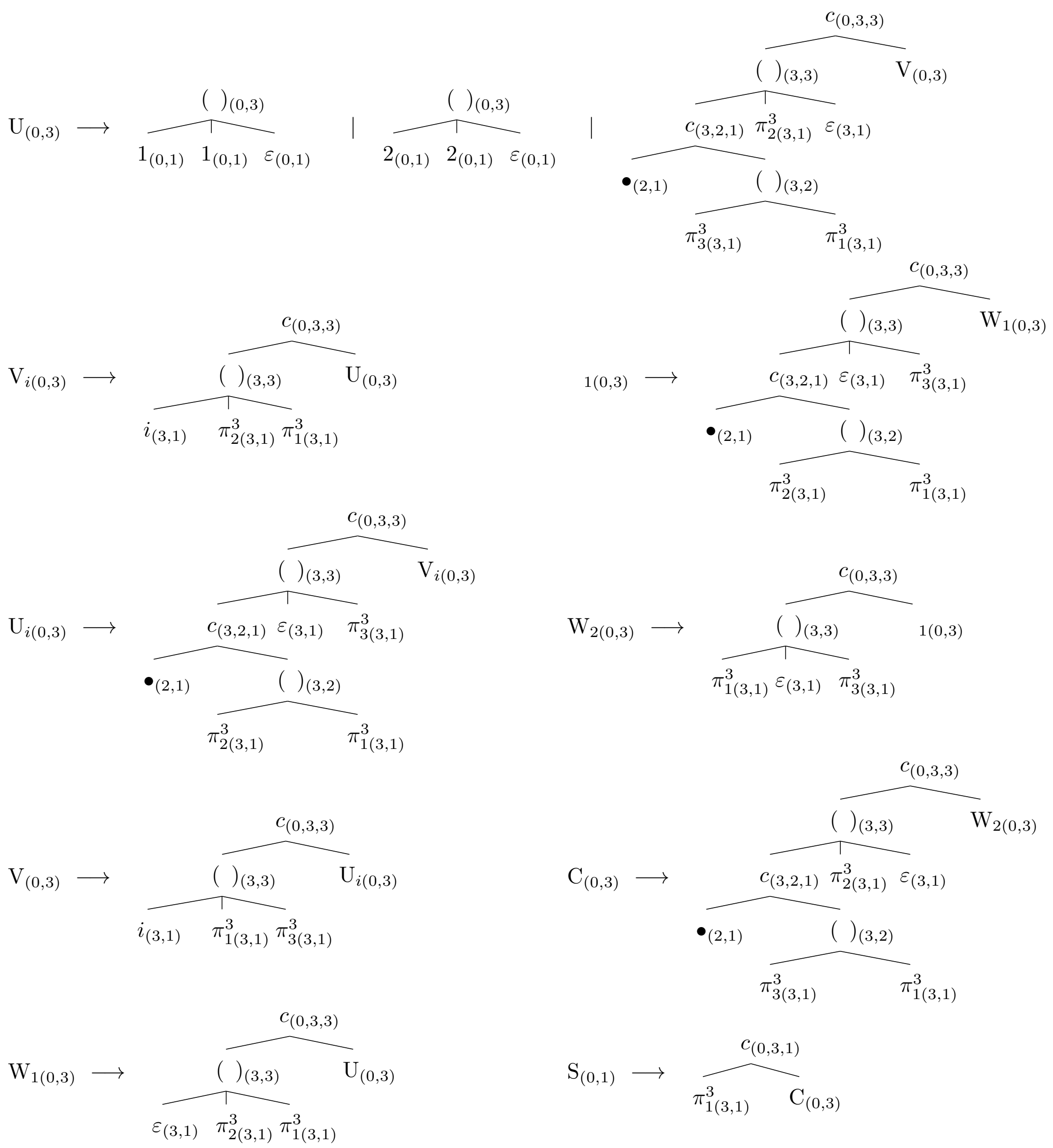
As one can see in Fig. 1, the basic functions have been realized as terms with their respective implicit operations as nonterminal (composition and tupling) or terminal (projection and empty tupling) nodes. In the following paragraphs, we sketch the translation T from nonterminal rules of the example MCFG to RTG rules. T takes each rule $X \longrightarrow f(Y)$, where $X, Y \in V_N$ and $f \in V_F$, of the MCFG including the corresponding definition of the mapping $f(x_1, \ldots, x_k)$ with $k \geq 0$ and transforms it into a RTG rule as follows. We create a mother node labeled with the appropriate binary composition $c_{(j,k,l)}$ such that the left daughter contains the "lifted" version of $f(x_1, \ldots, x_k)$ under T and the right daughter the translation of the nonterminal Y. Both nonterminals X and Y are used "unchanged", but annotated with the corresponding Lawvere arity resulting in the following schematic presentation of the translation: $X_{(j,l)} \longrightarrow c_{(j,k,l)}(\mathsf{T}(f(x_1, \ldots, x_k)), Y_{(j,k)})$, where f is a mapping from k-tuples to l-tuples of terminal strings.

The easiest case of translating a mapping $f \in F$ from our example via T is constituted by the terminal U-rules (4a). We simply view the mapping as a Lawvere term. The function $merge_{\langle\beta_i,\alpha_i\rangle}$ just returns a triple built from i, i and ε. The corresponding tree has a mother node labeled with a ternary tupling symbol and the three unary arguments of the mapping as daughters.[13] The nonterminating U-rule (4e) is more complicated: the function $move_{\langle\psi_{\text{-}}\rangle}$ takes three arguments, concatenates the last and first ones in the first argument slot, leaves the second slot untouched and inserts ε into the last one. The definition of the function can be written explicitly as the Lawvere term $(\ )_{(3,3)}(c_{(3,2,1)}(\bullet_{(2,1)}, (\ )_{(3,2)}(\pi^3_{3(3,1)}, \pi^3_{1(3,1)})), \pi^3_{2(3,1)}, \varepsilon_{(3,1)})$. Note again that the implicit binary concatenation $\bullet$ in $move_{\langle\psi_{\text{-}}\rangle}$ now becomes the constant $\bullet_{(2,1)}$. The variables are simply

[11] Note that we simplified the presentation of the rules at two points: $i_{(3,1)}$ and $\varepsilon_{(3,1)}$ represent trees resulting from lifting the "real" elements of $\Sigma_{\varepsilon,(0,1)}$ to the Lawvere arity $(3, 1)$, i.e., $i_{(3,1)}$ stands for $c_{(3,0,1)}(i_{(0,1)}, (\ )_{(3,0)})$ and $\varepsilon_{(3,1)}$ for $c_{(3,0,1)}(\varepsilon_{(0,1)}, (\ )_{(3,0)})$.

[12] For simplicity and readability we will sometimes drop the subscript notion (k, m) from the inoperatives and operatives of rank 0, and sometimes even from the composition symbol $c_{(k,l,m)}$.

[13] Note that we do not need to use a further composition symbol dominating $\mathsf{T}(f)$ in case there is no nonterminal on the RHS of the rule of the MCFG.



U(0,3) ⟶ ( )(0,3) 1(0,1) 1(0,1) ε(0,1) | ( )(0,3) 2(0,1) 2(0,1) ε(0,1) | c(0,3,3) ( )(3,3) V(0,3) c(3,2,1) π³2(3,1) ε(3,1) •(2,1) ( )(3,2) π³3(3,1) π³1(3,1)
Vi(0,3) ⟶ c(0,3,3) ( )(3,3) U(0,3) i(3,1) π³2(3,1) π³1(3,1)
1(0,3) ⟶ c(0,3,3) ( )(3,3) W1(0,3) c(3,2,1) ε(3,1) π³3(3,1) •(2,1) ( )(3,2) π³2(3,1) π³1(3,1)
Ui(0,3) ⟶ c(0,3,3) ( )(3,3) Vi(0,3) c(3,2,1) ε(3,1) π³3(3,1) •(2,1) ( )(3,2) π³2(3,1) π³1(3,1)
W2(0,3) ⟶ c(0,3,3) ( )(3,3) 1(0,3) π³1(3,1) ε(3,1) π³3(3,1)
V(0,3) ⟶ c(0,3,3) ( )(3,3) Ui(0,3) i(3,1) π³1(3,1) π³3(3,1)
C(0,3) ⟶ c(0,3,3) ( )(3,3) W2(0,3) c(3,2,1) π³2(3,1) ε(3,1) •(2,1) ( )(3,2) π³3(3,1) π³1(3,1)
W1(0,3) ⟶ c(0,3,3) ( )(3,3) U(0,3) ε(3,1) π³2(3,1) π³1(3,1)
S(0,1) ⟶ c(0,3,1) π³1(3,1) C(0,3)

Figure 1: The translated example grammar $\mathcal{G}'_{ww}$

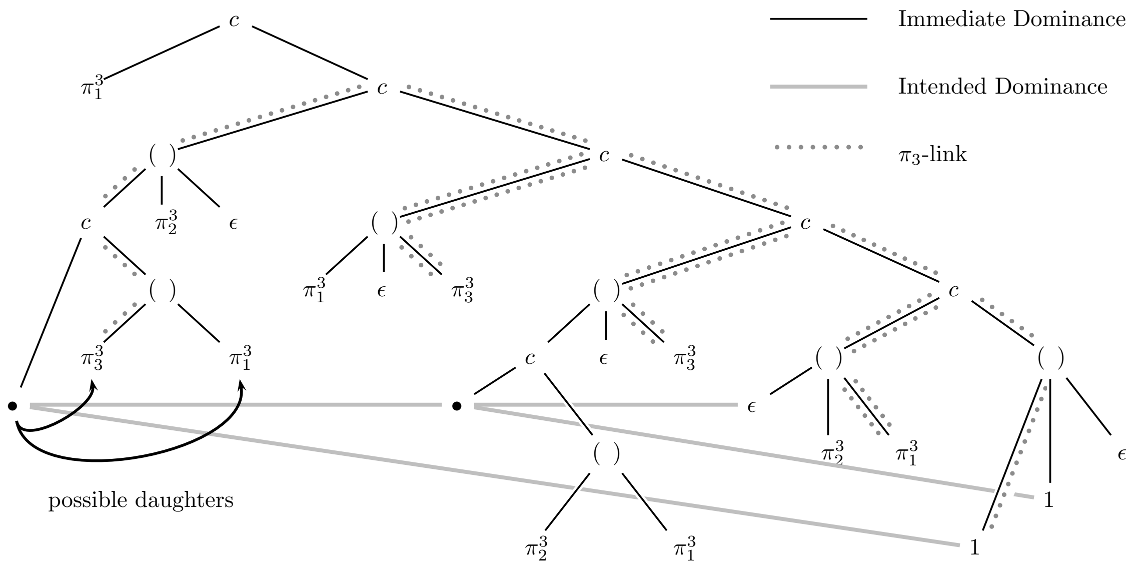


Figure 2: Reconstructing the Intended Structures

replaced by the projections and concatenated. The resulting term is then applied via composition to the operative $V_{(0,3)}$ such that we get the RHS displayed in the last disjunct of the $U_{(0,3)}$-rule in Fig. 1. A comparable procedure is followed with respect to the other MCFG-rules.

Since RTGs can only generate recognizable (tree) languages, we can characterize them with both MSO logic on trees and tree automata. The tree automaton $\mathfrak{A}_{\mathcal{G}'_{ww}}$ is constructed by transforming the grammar into a normal form such that each RHS is of depth one by introducing auxiliary operatives. Then we can easily construct appropriate transitions by basically reversing the arrow: the nonterminals become state names and the mother node will be read as alphabet symbol. It is know from Thomas (1990) how to transform this tree automaton into a Σ^1_1-formula $\varphi_{\mathfrak{A}_{\mathcal{G}'_{ww}}}$ by encoding its behaviour. In short, assuming an enumeration of $\mathfrak{A}$'s states $\{0, \ldots, m\}$ such that the initial state is represented by 0, the formula uses sets X_i to label the nodes where the automaton assumes state i. The first line of the formula says that we cannot have a node which is in two states and that X_0 is our "initial" set; the second one licenses the distribution of the sets according to the transitions and says that we need a root node which is in a "final" set. P_a stands for the predicate labeling a node with the symbol a. For n-ary tree automata the formula $\varphi_{\mathfrak{A}}$ looks as follows:

$$\begin{aligned}(\exists X_0, \ldots, X_m)[&\bigwedge_{i \neq j} (\neg\exists y)[y \in X_i \wedge y \in X_j] \wedge (\forall x)[(\neg\exists y)[x \lhd y] \to x \in X_0] \wedge \\ &(\forall x_1, \ldots, x_n, y)[\bigvee_{\substack{(i_1,\ldots,i_n,\sigma,j)\in\alpha \\ 1\leq k\leq n}} x_k \in X_{i_k} \wedge y \lhd x_k \wedge y \in X_j \wedge y \in P_\sigma] \bigvee_{i \in F} (\exists x \forall y)[x \lhd^* y \wedge x \in X_i]\end{aligned}$$

A more detailed description how to construct both the tree automaton and the corresponding MSO formula can be found in Kolb et al. (2000).

6 Reconstructing the Intended Trees

Unfortunately, the terminal tree in Fig. 2, generated/recognized by $\mathcal{G}'_{ww}$ given in Fig. 1, does not seem to have much in common with the structures linguists want to talk about.

Rogers (1998) has shown the suitability of an MSO description language for linguistics which is based upon the primitive relations of immediate ($\lhd$), proper ($\lhd^+$) and reflexive ($\lhd^*$) dominance and proper precedence ($\prec$). We will show how to define these relations with an MSO transduction thereby implementing the unique homomorphism mapping the terms into elements of the corresponding multiple context-free tree language, i.e., the trees linguists want to talk about.

The MSO transduction is in a certain sense universal, so we will not refer back to our particular example to present it. The only variable part is the number of different projection constants appearing in the RTG, i.e., π_1^3, π_3^3 and π_3^3.

At the core of the transduction is a tree-walking automaton defining the binary relation of immediate dominance ($\triangleleft$) on the nodes belonging to the intended structures. It is based on some simple observations. The reader is encouraged to check them against the example tree t' generated by $\mathcal{G}'_{ww}$ given in Fig. 2.

1. Our trees feature three families of labels: the "linguistic" symbols L, i.e., the lifted symbols of the underlying MCFG; the "composition" symbols $\mathsf{C} = \{c_{(u,v,w)}\}$; the "tupling" symbols $(\quad)_{(v,u)}$ and the "projection" symbols $\mathit{\Pi} = \{\pi_i^k\}$.

2. All nonterminal nodes in t' are labeled by some $c \in \mathsf{C}$ or a "tupling" symbol. Note that no terminal node is labeled by some c.

3. The terminal nodes in t' are either labeled by some "linguistic" symbol, a "tupling" symbol of the form $(\quad)_{(k,0)}$, i.e. the "empty" tuple, or by some "projection" symbol π_i^k.

4. Any "linguistic" node dominating anything in the intended tree is on some left branch in t', i.e., it is the left daughter of some $c \in \mathsf{C}$ and the sister of a tupling symbol whose daughters evaluate to the intended daughters.

5. For any node ν labeled with some "projection" symbol $\pi_i^u \in \mathit{\Pi}$ in t' there is a unique node μ (labeled with some $c \in \mathsf{C}$) which properly dominates ν and which immediately dominates a node labeled with a "tupling" symbol whose i-th daughter will eventually evaluate to the value of π_i^k. Moreover, μ will be the first node properly dominating ν which is on a left branch and bears a composition symbol. This crucial fact is arrived at by induction on the construction of $\mathcal{G}'_{ww}$ from $\mathcal{G}_{ww}$.

By 4. it is not hard to find possible dominees in any t' (e.g., the curved arrows in Fig. 2). It is the problem of determining the actual "filler" of a candidate-dominee which makes up the complexity of the definition of $\triangleleft$. There are three cases to account for:

1. If the node considered carries a "linguistic" label, it evaluates to itself;

2. if it has a "composition" label c, it evaluates to whatever its leftmost daughter evaluates to;

3. if it carries a "projection" label π_i^k, it evaluates to whatever the node it "points to"—by (5.) the i^{th} daughter of a "tupling" node which is dominated by the first C-node on a left branch dominating it—evaluates to.

Note that cases 2. and 3. are inherently recursive. In general, recursive definitions in MSO may lead to undecidability and are therefore disallowed. Fortunately, the use of tree-walking automata ensures the definability. In Fig. 2 the π_3^3-link is an example of such a path from the projection symbol to the corresponding filler.

According to the observations made above, a tree-walking automaton is defined to relate those nodes x and y which stand in the intended immediate dominance relation, i.e., $x \triangleleft y$. The automaton given graphically in Fig. 3. It starts on any node with a "linguistic" label (denoted here by L) which means for the given example $\bullet, 1, 2, \varepsilon$. Then it has to go up the first branch, read a composition symbol and descend to its sister. If it reads a "linguistic" node, the automaton stops. If it reads a composition symbol, the automaton goes to the left daughter and tries again. If it reads a tupling symbol, the automaton proceeds with its daughters (again, see the curved arrows in Fig. 2). On finding a projection symbol, it searches for the appropriate "filler" by going upwards until it is on a leftmost branch which is labeled with a composition symbol. Then it

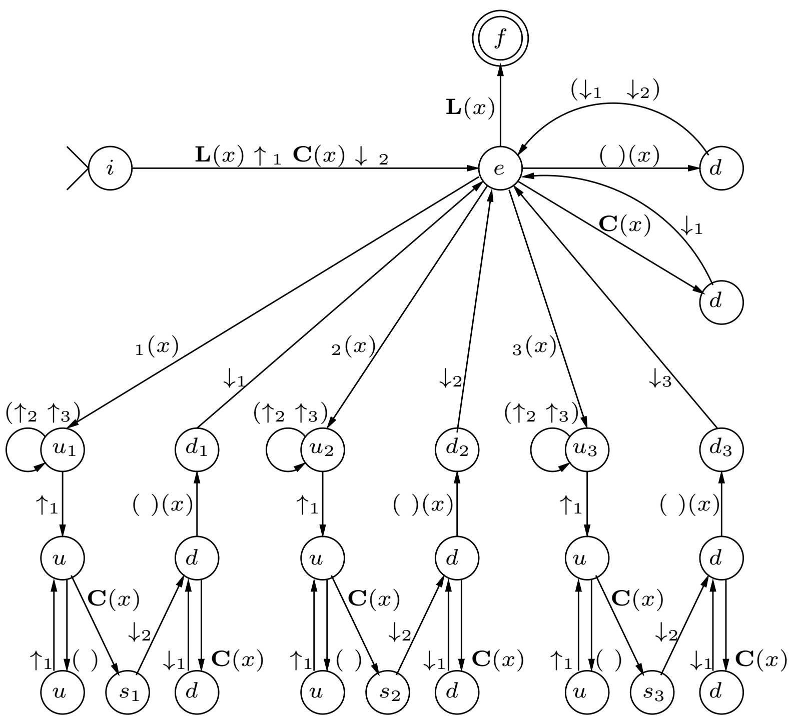


Figure 3: The tree-walking automaton for immediate dominance: $\mathfrak{A}_{\triangleleft}$

walks to the second sister or further down the leftmost branch until it hits a tupling node to whose appropriate daughter it descends to find the filler. The whole process is recursive, i.e., on finding another projection symbol, the automaton again tries to find an appropriate filler (see the π^3_3-link in Fig. 2.)

However, there is another interpretation of such an automaton. Viewed as an ordinary finite-state automaton over the alphabet Δ, $\mathfrak{A}_{\triangleleft}$ recognizes a regular (string-) language, the *walking language* $W_{\triangleleft} = \mathsf{L}(x) \cdot \uparrow_1 \cdot \mathsf{C}(x) \cdot \downarrow_2 \cdot (W_{(\ )} \cup W_{\mathsf{C}} \cup W_{\Pi_1} \cup W_{\Pi_2} \cup W_{\Pi_3})^* \cdot \mathsf{L}(x)$ with

$$\begin{array}{rcl}
W_{(\ )} & = & (\ )(x) \cdot (\downarrow_1 \cup \downarrow_2) \\
W_{\mathsf{C}} & = & \mathsf{C}(x) \cdot \downarrow_1 \\
W_{\Pi_1} & = & \Pi_1(x) \cdot (\uparrow_2 \cup \uparrow_3)^* \cdot \uparrow_1 \cdot ((\ )(x) \cdot \uparrow_1)^* \cdot \mathsf{C}(x) \cdot \downarrow_2 \cdot (\mathsf{C}(x) \cdot \downarrow_1)^* \cdot (\ )(x) \cdot \downarrow_1 \\
W_{\Pi_2} & = & \Pi_2(x) \cdot (\uparrow_2 \cup \uparrow_3)^* \cdot \uparrow_1 \cdot ((\ )(x) \cdot \uparrow_1)^* \cdot \mathsf{C}(x) \cdot \downarrow_2 \cdot (\mathsf{C}(x) \cdot \downarrow_1)^* \cdot (\ )(x) \cdot \downarrow_2 \\
W_{\Pi_3} & = & \Pi_3(x) \cdot (\uparrow_2 \cup \uparrow_3)^* \cdot \uparrow_1 \cdot ((\ )(x) \cdot \uparrow_1)^* \cdot \mathsf{C}(x) \cdot \downarrow_2 \cdot (\mathsf{C}(x) \cdot \downarrow_1)^* \cdot (\ )(x) \cdot \downarrow_3
\end{array}$$

which can be translated recursively into an MSO formula $\mathsf{trans}_{W_{\triangleleft}}$ defining the relation $\triangleleft$ (see Bloem and Engelfriet, 1997).

$$\begin{aligned}
\mathsf{trans}_{\emptyset}(x,y) &\equiv \bot \\
\mathsf{trans}_{\downarrow_i}(x,y) &\equiv \mathsf{edg}_i(x,y) \\
\mathsf{trans}_{\uparrow_i}(x,y) &\equiv \mathsf{edg}_i(y,x) \\
\mathsf{trans}_{\varphi(x)}(x,y) &\equiv \varphi(x) \wedge x = y \\
\mathsf{trans}_{W_1 \cup W_2}(x,y) &\equiv \mathsf{trans}_{W_1}(x,y) \vee \mathsf{trans}_{W_2}(x,y) \\
\mathsf{trans}_{W_1 \cdot W_2}(x,y) &\equiv (\exists z)[\mathsf{trans}_{W_1}(x,z) \wedge \mathsf{trans}_{W_2}(z,y)] \\
\mathsf{trans}_{W^*}(x,y) &\equiv \mathsf{trans}^*_W(x,y) \\
\mathsf{trans}^*_W(x,y) &\equiv (\forall X)(\forall v,w)[(v \in X \wedge \mathsf{trans}_W(v,w) \to w \in X) \wedge x \in X \to y \in X]
\end{aligned}$$

where the edge relation is defined as follows

$$\begin{aligned}\mathsf{edg}_n(x,y) \overset{def}{\iff} (\exists x_1,\dots,x_{n-1})[x \lhd x_1 \land \cdots \land x \lhd x_{n-1} \land x \lhd y \land x_1 \prec x_2 \land \cdots \land x_{n-1} \prec y \\ \land (\forall w)[x \lhd w \land w \not\approx x_1 \land \cdots \land w \not\approx x_{n-1} \land w \not\approx y \to y \prec w]]\end{aligned}$$

and $\varphi(x)$ in $\mathsf{trans}_{\varphi(x)}(x,y)$ stands in our case for the tests $(\ )(x)$, $\mathsf{C}(x)$ and $\mathsf{L}(x)$. We leave the rather tedious process of converting the walking language for the automaton given in Fig. 3 to the reader (a full example of such a conversion can be found in Kolb et al. 2000).

To present the actual MSO transduction, we need one further auxiliary definition. It is a well-known fact (e.g. Bloem and Engelfriet 1997) that the reflexive transitive closure R^* of a binary relation R on nodes is (weakly) MSO-definable, if R itself is. This is done via a second-order property which holds of the sets of nodes which are closed under R: $R{-}\mathsf{closed}(X) \iff_{def} (\forall x,y)[x \in X \land R(x,y) \to y \in X]$.

Finally, the MSO transduction $(\varphi, \psi, (\theta_q)_{q\in Q})$ with $Q = \{\lhd, \lhd^*, \lhd^+, \prec, \dots\}$ we use to transform the lifted structures into the intended ones is given as follows:

$$\begin{aligned}\varphi &\equiv \varphi_{\mathfrak{A}_{\mathcal{G}'_{ww}}}\\ \psi &\equiv (\exists y)[\mathsf{trans}_{W_\lhd}(x,y) \lor \mathsf{trans}_{W_\lhd}(y,x)]\\ \theta_\lhd(x,y) &\equiv \mathsf{trans}_{W_\lhd}(x,y)\\ \theta_{\lhd^*}(x,y) &\equiv (\forall X)[\lhd{-}\mathsf{closed}(X) \land x \in X \to y \in X]\\ \theta_{\lhd^+}(x,y) &\equiv x \lhd^* y \lor x \not\approx y\\ \theta_\prec(x,y) &\equiv \text{another tree-walking automaton}\\ \theta_{\text{labels}} &\equiv \text{taken over from } R\end{aligned}$$

As desired, the domain of the transduction is characterized by the MSO formula $\varphi_{\mathfrak{A}_{\mathcal{G}'_{ww}}}$ for the lifted trees. The domain, i.e., the set of nodes, of the intended tree is characterized by the formula ψ which identifies the nodes with a “linguistic” label which stand indeed in the new dominance relation to some other node. Building on it, we define the other primitives of a tree description language suited to linguistic needs. For reasons of space, we have to leave the specification of the precedence relation open. It is more complicated than dominance, but can be achieved with another tree-walking automaton.

7 Conclusion

Taking the result of Michaelis' translation of MGs as the input we have shown how to define a RTG by lifting the corresponding MCFG-rules by viewing them as terms of a free Lawvere theory. This gives us both a regular (via tree and tree-walking automata) and a logical characterization (via MSO logic and an MSO definable transduction) of the intended syntactic trees. Equivalently, we provide both an operational and a denotational account of Stabler's version of Minimalism without having to go via derivation trees.

It remains to be seen whether one can find a machine model for the entire MSO transduction. A likely candidate are the macro tree transducers (MTT) introduced in Engelfriet and Maneth (1999). Since they characterize the class of MSO definable tree translations if extended with regular look-ahead and restricted to finite-copying, we are quite optimistic that we will be able to use them to efficiently implement the transduction. This would also characterize the class of languages we can handle. Engelfriet and Maneth show that the result of applying MTTs to a regular tree languages yields the tree languages generated by context-free graph grammars.

Acknowledgements

The research presented in this paper was supported by the *Deutsche Forschungsgemeinschaft* within the *Sonderforschungsbereich* 340, TP A8. The authors wish to thank Kai-Uwe Kühnberger and Stephan Kepser for helpful discussions.

References

Bloem, R. and Engelfriet, J. (1997). Characterization of properties and relations defined in monadic second order logic on the nodes of trees. Technical Report 97-03, Leiden University.

Courcelle, B. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In Rozenberg, G., editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*, chapter 5, pages 313–400. World Scientific.

Doner, J. E. (1970). Tree acceptors and some of their applications. *J. Comput. System Sci.*, 4:406–451.

Engelfriet, J. (1997). Context-free graph grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages. Vol. III: Beyond Words*, chapter 3, pages 125–213. Springer.

Engelfriet, J. and Maneth, S. (1999). Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, 154:34–91.

Gécseg, F. and Steinby, M. (1984). *Tree Automata.* Akadémiai Kiadó, Budapest.

Kolb, H.-P., Mönnich, U., and Morawietz, F. (2000). Descriptions of cross-serial dependencies. To appear in a special issue of *Grammars*. Draft available under `http://tcl.sfs.nphil.uni-tuebingen.de/~frank/`.

Mezei, J. and Wright, J. (1967). Algebraic automata and contextfree sets. *Information and Control*, 11:3–29.

Michaelis, J. (1999). Derivational minimalism is mildly context-sensitive. In Moortgat, M., editor, *LACL '98*, LNAI. Springer. To appear.

Mönnich, U. (1998). TAGs M-constructed. In *TAG+ 4th Workshop, Philadelphia.*

Rabin, M. O. (1969). Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35.

Rambow, O. and Satta, G. (1999). Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223(1–2):87–120.

Rogers, J. (1998). *A Descriptive Approach to Language-Theoretic Complexity.* Studies in Logic, Language, and Information. CSLI Publications and FoLLI.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Stabler, E. (1997). Derivational minimalism. In Retoré, C., editor, *Logical Aspects of Computational Linguistics*, pages 68–95, Berlin. Springer. LNAI 1328.

Stabler, E. (1999). Remnant movement and complexity. In Bouma, G., Kruijff, G.-J. M., Hinrichs, E., and Oehrle, R. T., editors, *Constraints and Resources in Natural Language Syntax and Semantics*, volume II of *Studies in Constrained Based Lexicalism*, pages 299–326. CSLI.

Thatcher, J. W. and Wright, J. B. (1968). Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81.

Thomas, W. (1990). Automata on infinite objects. In van Leeuwen, J., editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V.

Wagner, E. G. (1994). Algebraic semantics. In Abramsky, S., Gabbay, D. M., and Maibaum, T. S. E., editors, *Semantic Structures*, volume 3 of *Handbook of Logic in Computer Science*, pages 323–393. Oxford University Press.

Weir, D. J. (1992). Linear context-free rewriting systems and deterministic tree-walk transducers. In *30th Meeting of the Association for Computational Linguistics (ACL'92).*