Electronic Notes in Theoretical Computer Science 53 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume53.html 23 pages

A Model-Theoretic Description of Tree Adjoining Grammars [1]

Frank Morawietz and Uwe Mönnich [2,3]

Seminar für Sprachwissenschaft
Universität Tübingen
Tübingen, Germany

Abstract

In this paper we show that non-context-free phenomena can be captured using only limited logical means. In particular, we show how to encode a Tree Adjoining Grammar [16] into a weakly equivalent monadic context-free tree grammar (MCFTG). By viewing MCFTG-rules as terms in a free Lawvere theory, we can translate a given MCFTG 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 through a model-theoretic interpretation in the form of an MSO transduction based upon tree-walking automata. This two-step approach gives a logical as well as an operational 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.

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

[1] The research presented in this paper was supported by the *Deutsche Forschungsgemeinschaft* within the *Sonderforschungsbereich* 441, TP A2. The authors wish to thank Jens Michaelis and Stephan Kepser for helpful comments.

[2] Email: `frank@sfs.uni-tuebingen.de`
[3] Email: `um@sfs.uni-tuebingen.de`

For regular string and tree languages, classical results in the descriptive theory of recognizability have established a tight connection between logical formalisms and language classes. They provide translation procedures that transform logical specifications into finite automata equivalent to the language classes and vice versa. Büchi [3] and Elgot [7] have shown that regular string languages represented through finite (string) automata can be expressed by sentences in the weak MSO logic with one successor. For tree languages an analogous result is well known: a tree language is definable in weak MSO logic with multiple successors if and only if it is recognizable by a finite tree automaton [6,32].

It is these earlier characterizations that provide the reason for a renewed interest in logical approaches to grammar specifications. The main open question in this area of research is whether an appropriate extension of the MSO language can be found which is expressive enough to define significant properties of natural languages without becoming too unwieldy from the perspective of complexity theory.

We believe, similarly to the claims in Pullum and Scholz [28], that model-theoretic approaches to syntax represent an important aspect of the formal analysis of natural language formalisms. In the present context we try to circumvent the problems posed by the limited expressive power of MSO by developing the model-theoretic description in terms of a two-step approach. We choose Tree Adjoining Grammar (TAG) [13,14,15] as maybe the prototypical mildly context-sensitive formalism [4] for natural languages to illustrate our proposal for a denotational approach towards a generative, non-context-free formalism.

The first step in the approach proposed in this paper consists in the use of a type of tree grammars which can be "lifted", i.e., a certain amount of control information is explicitly coded in the trees, such that the resulting tree sets are amenable to formalizations in terms of tree automata and MSO logic. In particular, we will use a restricted form of context-free tree grammars (CFTGs), namely *monadic* CFTGs which have an adequate descriptive complexity (see Section 2.3 below). Since the weak equivalence between TAGs and monadic CFTGs has independently been proven by Mönnich [24] and Fujiyoshi and Kasai [12], CFTGs form an adequate basis for the following work. Finally, a second step is necessary to remove the explicit control information from the trees to recover the linguistically intended structures.

Recently, Jim Rogers has shown that tree-adjoining languages can be characterized in terms of a monadic second-order definition on the three-dimensional tree manifolds [30]. We plan to work out in a future paper the exact relationship between the introduction of a further tree dimension and the device of higher types in the specification of lifted vocabularies.

[4] In fact, the desiderata Joshi enumerates for any formalism dealing with natural languages coined this terminology.

2 Preliminaries

2.1 Universal Algebra

Recall that for a given set of sorts $\mathcal{S}$, a *many-sorted alphabet* 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 $\sigma \in \Sigma_{w,s}$ is an *operator of type* $\langle w, s \rangle$, *arity* w, *sort* s and *rank* $|w|$. The elements of $\Sigma_{\varepsilon,s}$ are also called constants (of sort s).

In case $\mathcal{S}$ is a singleton set $\{s\}$, i.e., in case Σ is a *single-sorted or ranked alphabet (over sort* s*)*, we usually write Σ_n to denote the (unique) set of operators of rank $n \in \mathbb{N}$.[5]

In later sections of the paper we will mainly use the single-sorted case of alphabets. We will indicate the need for many-sorted alphabets where necessary.

For such a ranked alphabet Σ, we denote by $T(\Sigma)$ the set of *trees* over Σ. $T(\Sigma)$ is inductively defined with base case $\Sigma_0 \subseteq T(\Sigma)$ and recursive step $f(t_1, \ldots, t_n) \in T(\Sigma)$ if $f \in \Sigma_n$ and $t_i \in T(\Sigma)$ for $i = 1, \ldots, n$.

Furthermore, we fix an indexed set $X = \{x_1, x_2, \ldots\}$ of *variables* and denote by X_n the subset $\{x_1, \ldots, x_n\}$. Variables are considered to be constants, i.e., operators of rank 0. For a ranked alphabet Σ the family $T(\Sigma, X)$ is defined to be $T(\Sigma(X))$, where $\Sigma(X)$ is the ranked alphabet with $\Sigma(X)_0 = \Sigma_0 \cup X$ and $\Sigma(X)_n = \Sigma_n$ for every $n \neq 0$. A subset L of $T(\Sigma)$ is called a *tree language* over Σ.

Having described the tree terms, it remains to specify the central notion of an algebra and to give a precise definition of the way in which the operator symbols induce operations on an algebra.

Suppose that Σ is a ranked alphabet. A Σ-*algebra* $\mathbb{A}$ is a pair $\mathbb{A} = (A, (f_{\mathbb{A}})_{f \in \Sigma})$ where the set A is the *carrier* of the algebra and for each operator $f \in \Sigma_n$, $f_{\mathbb{A}} : A^n \rightarrow A$ is an operation of arity n on A.

Different algebras, defined over the same operator domain, are related to each other if there exists a mapping between their carriers that is compatible with the basic structural operations.

A Σ-*homomorphism* of Σ-algebras $h : \mathbb{A} \longrightarrow \mathbb{B}$ is a function $h : A \longrightarrow B$, such that $h(f_{\mathbb{A}}(a_1, \ldots, a_n)) = f_{\mathbb{B}}(h(a_1), \ldots, h(a_n))$ for every operator f of rank n and for every n-tuple $(a_1, \ldots, a_n) \in A^n$.

The set of trees $T(\Sigma, X)$ can be made into a Σ-algebra $\mathbb{T}(\Sigma, X)$ by defining the operations in the following way. For every f in Σ_n, for every $(t_1, \ldots, t_n)$ in $T(\Sigma, X)^n$: $f_{\mathbb{T}(\Sigma,X)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.

Every variable-free tree $t \in T(\Sigma)$ has a value in every Σ-algebra $\mathbb{A}$. It is the value at t of the unique homomorphism $h : \mathbb{T}(\Sigma) \rightarrow \mathbb{A}$.

The existence of a unique homomorphism from the Σ-algebra of trees into an arbitrary Σ-algebra $\mathbb{A}$ provides also the basis for the view that regards the

[5] 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 $n \in \mathbb{N}$.

elements of $T(\Sigma, X_n)$ as *derived operations*. Each tree $t \in T(\Sigma, X_n)$ induces an n-ary function $t_{\mathbb{A}} : A^n \to A$.

The meaning of this function $t_{\mathbb{A}}$ is defined in the following way. For every $(a_1, \dots, a_n) \in A^n$: $t_{\mathbb{A}}(a_1, \dots, a_n) = \hat{a}(t)$, where $\hat{a} : \mathbb{T}(\Sigma, X_n) \to \mathbb{A}$ is the unique homomorphism with $\hat{a}(x_i) = a_i$.

In the particular case where $\mathbb{A}$ is the Σ-algebra $\mathbb{T}(\Sigma, X_m)$ of trees over Σ that contain at most variables from $X_m = \{x_1, \dots, x_m\}$ at their leaves the unique homomorphism extending the assignment of a tree $t_i \in T(\Sigma, X_m)$ to the variable x_i in X_n acts as a substitution $t_{\mathbb{T}(\Sigma, X_m)}(t_1, \dots, t_n) = t[t_1, \dots, t_n]$ where the right hand side indicates the result of substituting t_i for x_i in t.

2.2 Tree Adjoining Grammar

The main obstacle for the simple and direct use of MSO logic stems from the fact that linguistic theories allow non-context-free structures and therefore cannot be accommodated within the classical approach outlined in Rogers' monograph [29]. In this section we will sketch a definition of tree adjoining grammars following Vijay-Shanker and Weir [34] and Joshi and Schabes [16]. This formalism has a descriptive complexity which is higher than that of context-free grammars. Additionally, TAGs are a derivational formalism and, hence, a logical approach towards describing the resulting tree sets presents a further problem.

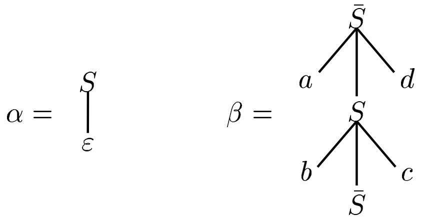
Definition 2.1 (Tree Adjoining Grammar) *A Tree Adjoining Grammar (TAG) is a quintuple $\langle V_N, V_T, S, \mathcal{I}, \mathcal{A} \rangle$ where V_N is a finite set of nonterminals, V_T a finite set of terminals, $S \in V_N$ the start symbol, $\mathcal{I}$ a finite set of initial trees and $\mathcal{A}$ a finite set of auxiliary trees.*

Initial trees are such that all interior nodes (including the root node) are labeled with nonterminals and all nodes on the frontier are labeled with terminal or nonterminal symbols; the nonterminals being marked for substitution. The same holds for the auxiliary trees with one exception. There exists one distinguished leaf-node which is labeled with the same nonterminal as the root node, which is called the foot node. Furthermore, the nodes can be marked to allow or forbid adjunction. For simplicity, we only indicate nodes where no adjunction is allowed by putting a bar on top of them.

New trees are built from the sets $\mathcal{I}$ and $\mathcal{A}$ via adjunction or substitution. Adjunction is defined such that an auxiliary tree is spliced into an existing tree such that it basically "expands" a nonterminal. A subtree rooted in the node labeled with a nonterminal A is taken out of the a tree. A new auxiliary tree is inserted in its place (if the root and foot are also labeled with the identical nonterminal A) and the original subtree is appended at the foot node. There also exists a simpler operation in TAGs, called *substitution*, to generate new trees. Intuitively, in substitution, a nonterminal is replaced by a tree with a matching nonterminal at its root. Since we do not need the formal definitions in this paper, the reader is referred to the literature cited above for details.

The corresponding tree and string languages are defined in the obvious way. An example for a TAG generating the non-CF language $a^nb^nc^nd^n$ is given below:

Example 2.2 *Let* $G_{TAG} = \langle \{S\}, \{a,b,c,d\}, S, \{\alpha\}, \{\beta\} \rangle$ *be a TAG. The only initial tree* α *and the only auxiliary tree* β *are given as follows:*



A derivation yielding *aabbccdd* has only two steps, both adjoin the auxiliary tree in the only possible position, see Figure 1.

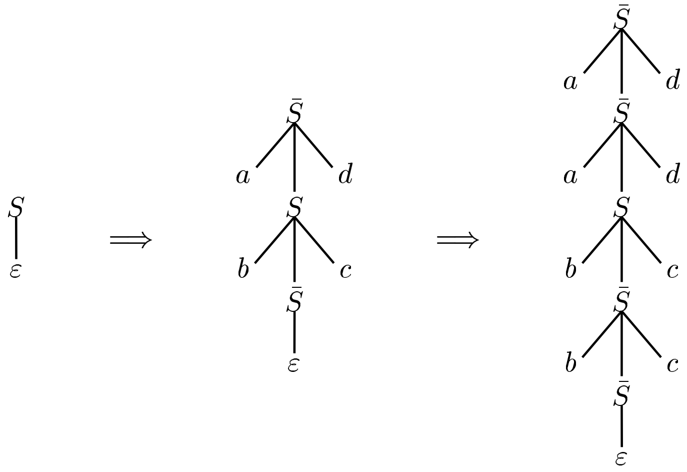


Fig. 1. An example derivation of the TAG G_{TAG} given in Example 2.2

It can be shown that TAGs can only generate string languages with dependencies up to four, i.e., $a^nb^nc^nd^n$ can be generated, but there is no TAG which generates $a^nb^nc^nd^ne^n$ [16].

2.3 Context-Free Tree Grammars

The algebraic perspective allows the uniform and natural extension from strings to trees by the simple technique of generalizing from unary to multiary operators. In this section, grammars working on strings are generalized to grammars working on trees. Thus we have a natural counterpart to the Chomsky hierarchy: regular tree grammars correspond to the known regular grammars and context-free tree grammars to the context-free grammars.

We now formally introduce the notion of a context-free tree grammar (CFTG). This type of grammar is related to a type of grammars which were defined by Fisher [10] and which were called *macro grammars*. In his setting, the use of macro-like productions served the purpose of making simultaneous string copying a primitive operation. CFTGs constitute an algebraic generalization of macro grammars (cf. Rounds [31]).

Let us view grammars as a mechanism in which local transformations on trees can be performed. The central ingredient of a grammar is a finite set of productions, where each production is a pair of trees. Such a set of productions determines a binary relation on trees such that two trees t and t' stand in that relation if t' is the result of removing in t an occurrence of a first component in a production pair and replacing it by the second component of the same pair. The simplest type of such a replacement is defined by a production that specifies the substitution of a single-node tree t_0 by another tree t_1. Two trees t and t' satisfy the relation determined by this simple production if the tree t' differs from the tree t in having a subtree t_1 that is rooted at an occurrence of a leaf node t_0 in t. In slightly different terminology, productions of this kind incorporate instructions to rewrite auxiliary variables as a complex symbol that, autonomously, stands for an element of a tree algebra. Recall that in context-free string grammars a nonterminal auxiliary symbol is rewritten as a string of terminal and nonterminal symbols, independently of the context in which it occurs. As long as the carrier of a tree algebra is made of constant tree terms the process of replacing null-ary variables by trees is analogous. As we will see, the situation changes dramatically if the carrier of the algebra is made of symbolic counterparts of derived operations and the variables in production rules range over these second-level entities.

Definition 2.3 (Context-Free Tree Grammar) *Let $\mathcal{S}$ be a singleton set of sorts. Then a* context-free tree grammar (CFTG) *for $\mathcal{S}$ is a 5-tuple $\Gamma = \langle \Sigma, \mathsf{F}, S, \mathsf{X}, \mathsf{P} \rangle$, where Σ and F are ranked alphabets of* inoperatives *and* operatives *over $\mathcal{S}$, respectively. $S \in \mathsf{F}$ is the start symbol, X is a countable set of variables, and P is a set of productions. Each $p \in \mathsf{P}$ is of the form $F(x_1, \cdots, x_n) \longrightarrow t$ for some $n \in \mathbb{N}$, where $F \in \mathsf{F}_n$, $x_1, \cdots, x_n \in \mathsf{X}$, and $t \in T(\Sigma \cup \mathsf{F}, \{x_1, \cdots, x_n\})$.*

Intuitively, an application of a rule $F(x_1, \ldots, x_n) \rightarrow t$ "rewrites" a tree rooted in F as the tree t with its respective variables substituted by F's daughters.

A CFTG $\Gamma = \langle \Sigma, \mathsf{F}, S, \mathsf{X}, \mathsf{P} \rangle$ with $\mathsf{F}_n = \emptyset$ for $n \neq 0$ is called a *regular tree grammar (RTG)*. Since RTGs always just substitute some tree for a leaf-node, it is easy to see that they can only generate recognizable sets of trees, *a forteriori* context-free string languages [22]. If F_n is non-empty for some $n \neq 0$, that is, if we allow the *operatives* to be parameterized by variables, however, the situation changes. CFTGs in general are capable of generating sets of structures, the *yields* of which belong to the subclass of context-sensitive languages known as the *indexed* languages. In fact, CFTGs characterize the class of indexed languages modulo the inside-out derivation mode [31].

For reasons having to do with the impossibility of mirroring the process of copying in a grammar with a completely uncontrolled derivation regime, we restrict ourselves to this particular mode of derivation. Accordingly, a function symbol may be replaced only if all its arguments are trees over the terminal alphabet. In the conventional case this form of replacement mechanism would correspond to a "rightmost" derivation where "rightmost" is to be understood with respect to the tree order.

The definition of a CFTG given above could be canonically generalized to the case of many-sorted signatures Σ and F over some set of sorts $\mathcal{S}$. Since we will be concerned with such generalized versions of CFTGs only in their regular form, we will not give a new definition but rely on the definition of a many-sorted alphabet in Section 2.1 and reader's intuition.

Finally, since we need to show how to handle TAGs within the framework of CFTGs, we also define monadic context-free tree grammars (MCFTGs). Since they are only a simplification of Definition 2.3, we outline only the differences here.

Definition 2.4 (Monadic Context-Free Tree Grammar) *Given a set of sorts $\mathcal{S}$, a monadic context-free tree grammar (MCFTG) for a set of sorts $\mathcal{S}$ is a 5-tuple $\Gamma = \langle \Sigma, \mathsf{F}_0 \cup \mathsf{F}_1, S, \mathsf{X}, \mathsf{P}\rangle$, i.e., a CFTG, where all the rules in P are of one of the following "unary" types ($A, B, C, B_i \in \mathsf{F}_1 \cup \mathsf{F}_0$, $1 \leq i \leq n$, $a \in \Sigma$, $x \in \mathsf{X}$):*

$$\begin{aligned} A &\longrightarrow a \\ A &\longrightarrow B(C) \\ A(x) &\longrightarrow a(B_1, \ldots, B_{i-1}, x_i, B_{i+1}, \ldots, B_n) \\ A(x) &\longrightarrow B_1(B_2(\ldots B_n(x)\ldots)) \end{aligned}$$

Interestingly, it has been shown by Mönnich [24] and Fujiyoshi and Kasai [12] that TAGs are weakly equivalent to this restricted form of CFTGs. To a large extend, the intuition behind the proof is fairly simple. Since CFTGs can insert multiple subtrees in a single step, but TAGs only a single one, all we have to do is limit the operative nonterminals of the CFTG to unary or monadic ones.

Since the proof is constructive, i.e., there exists an algorithm to transform any given TAG into an equivalent monadic CFTG, we will silently assume in the following sections that it is enough to deal with (M)CFTGs.

As an example, we present an MCFTG Γ_{TAG} for $a^n b^n c^n d^n$ generating the same language as the TAG grammar G_{TAG} given in Example 2.2.

Example 2.5 *Consider the MCFTG $\Gamma_{TAG} = \langle\{a, b, c, d, \varepsilon, S_t, S_t^0\}, \{S, S', \overline{S}_1, \overline{S}_2, \overline{a}, \overline{b}, \overline{c}, \overline{d}\}, S', \{x\}, \mathsf{P}\rangle$ resulting from a translation of the TAG G_{TAG} with P given as follows*

$$\begin{aligned} S' &\longrightarrow S(\varepsilon) & \overline{a} &\longrightarrow a \\ S(x) &\longrightarrow \overline{S}_1(S(\overline{S}_2(x))) & \overline{b} &\longrightarrow b \\ S(x) &\longrightarrow S_t^0(x) & \overline{c} &\longrightarrow c \\ \overline{S}_1(x) &\longrightarrow S_t(\overline{a}, x, \overline{d}) & \overline{d} &\longrightarrow d \\ \overline{S}_2(x) &\longrightarrow S_t(\overline{b}, x, \overline{c}) \end{aligned}$$

A corresponding derivation of the string *abbccdd* is shown in Figure 2. The example derivation is somewhat longer than the one given for the almost identical TAG grammar generating the same language. This is due to the fact that we need nonterminals to introduce each branching of the resulting tree separately. In the first step, we simply rewrite the start symbol. In the second one, the symbol S with the term $\overline{S}_1(S(\overline{S}_2(x)))$ where the (degenerate) tree ε is simply appended to the only argument position x of $\overline{S}_2$. This step is repeated before we terminate with an application of the rule rewriting S as S_t^0. We simplified the presentation in the sense that in this last step we also applied the rules for the "barred" operatives, i.e., we replaced each $\overline{S}_i$, $i \in \{1, 2\}$ with the corresponding term and each $\overline{s} \in \{\overline{a}, \overline{b}, \overline{c}, \overline{d}\}$ with s.

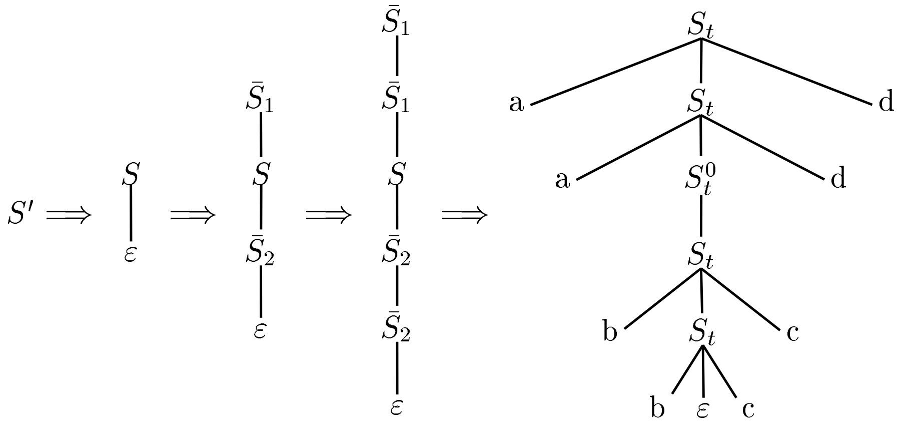


Fig. 2. An example derivation of the MCFTG Γ_{TAG} from Example 2.5

3 Lifting

We designate the process which makes the control information inherent in term based grammar formalisms explicit by the term LIFTing. The intuition here is that the basic assumptions about the operations of tree substitution and argument insertion are made explicit. We make them visible by inserting the "control" information which allows us to code the resulting structures with regular means, i.e., regular tree grammars or finite-state tree automata and therefore with MSO logic.

3.1 LIFT*ing CFTGs*

The intuition behind the LIFTing process is that each term compactly encodes information such as composition and concatenation.

Any *context-free* tree grammar Γ for a singleton set of sorts $\mathcal{S}$ can be transformed into a *regular* tree grammar Γ^L for the set of sorts $\mathcal{S}^*$, which characterizes a (necessarily recognizable) set of trees encoding the instructions necessary to convert them by means of a unique homomorphism h into the ones the original grammar generates [21]. This "LIFTing" is achieved by constructing for a given single-sorted signature Σ a new, derived alphabet (an $\mathbb{N}$-sorted signature) Σ^L, and by translating the terms over the original signature into terms of the derived one via a primitive recursive procedure. The LIFT-operation takes a term in $T(\Sigma, \mathsf{X}_k)$ and transforms it into one in $T(\Sigma^L, k)$.[6] Intuitively, the LIFTing eliminates variables and composes functions with their arguments explicitly, e.g., a term $f(a, b) = f(x_1, x_2) \circ (a, b)$ is lifted to the term $c(c(f, \pi_1, \pi_2), a, b)$. The old function symbol f now becomes a constant, the variables are replaced with appropriate projection symbols and the only remaining non-nullary alphabet symbols are the explicit composition symbols c.

Definition 3.1 (lift) *Let Σ be a ranked alphabet and $\mathsf{X}_k = \{x_1, \ldots, x_k\}$, $k \in \mathbb{N}$, a finite set of variables. The* derived $\mathbb{N}$-*sorted alphabet* Σ^L *is defined as follows: For each $n \geq 0$, $\Sigma'_{\varepsilon,n} = \{f' \mid f \in \Sigma_n\}$ is a new set of symbols of type $\langle \varepsilon, n\rangle$; for each $n \geq 1$ and each $i, 1 \leq i \leq n$, π^n_i is a new symbol, the ith* projection symbol *of type $\langle \varepsilon, n\rangle$; for each $n, k \geq 0$ the new symbol $c_{(n,k)}$ is the (n,k)th* composition symbol *of type $\langle nk_1 \cdots k_n, k\rangle$ with $k_1 = \cdots = k_n = k$. The set of all $c_{(n,k)}$ will be denoted by C, the set of all π^n_i by Π.*

$$\begin{aligned}
\Sigma^L_{\varepsilon,0} &= \Sigma'_{\varepsilon,0} \\
\Sigma^L_{\varepsilon,n} &= \Sigma'_{\varepsilon,n} \cup \{\pi^n_i \mid 1 \leq i \ \leq n\} \ \ \textit{for} \ \ n \geq 1 \\
\Sigma^L_{nk_1\cdots k_n,k} &= \{c_{(n,k)}\} \ \ \textit{for} \ \ n, k \geq 0 \ \ \textit{and} \ \ k_i = k \ \ \textit{for} \ \ 1 \leq i \leq k \\
\Sigma^L_{w,s} &= \emptyset \ \ \textit{otherwise}
\end{aligned}$$

For $k \geq 0$, $\text{LIFT}^\Sigma_k : T(\Sigma, \mathsf{X}_k) \to T(\Sigma^L, k)$ is defined as follows:

$$\begin{aligned}
\text{LIFT}^\Sigma_k(x_i) &= \pi^k_i \\
\text{LIFT}^\Sigma_k(f) &= c_{(0,k)}(f') \ \ \textit{for} \ \ f \in \Sigma_0 \\
\text{LIFT}^\Sigma_k(f(t_1, \ldots, t_n)) &= c_{(n,k)}(f', \text{LIFT}^\Sigma_k(t_1), \ldots, \text{LIFT}^\Sigma_k(t_n)) \\
&\qquad \textit{for} \ \ n \geq 1, f \in \Sigma_n \ \ \textit{and} \ \ t_1, \ldots, t_n \in T(\Sigma, X_k)
\end{aligned}$$

Note that this very general procedure allows the translation of any term over the original signature. The left hand side as well as the right hand side (RHS) of a rule of a CFTG $\Gamma = \langle \Sigma, \mathsf{F}, \mathsf{X}, S, \mathsf{P}\rangle$ is just a term belonging to $T(\Sigma \cup \mathsf{F}, \mathsf{X})$, but so is, e.g., any structure *generated* by Γ.

Further remarks on the observation that the result of LIFTing a CFTG is always an RTG can be found in Mönnich [25].

[6] Since $\mathcal{S}$ is a singleton, we can identify S^* with $\mathbb{N}$. By $T(\Sigma^L, k)$ we denote the set of all trees over Σ^L which are of sort k.

Lifting MCFTGs uses the same definitions for LIFTing as is used for general CFTGs. The translation process for grammars has at its heart the LIFT-morphism for the translation of the alphabets of the operatives and inoperatives and the RHSs of the production rules. Since the rest of the translation follows trivially from this, we dispense with a formal definition. Note that for better readability, we omit all the 0- and 1-place composition symbols. To further illustrate the techniques, we present the continuation of Example 2.5.

Example 3.2 *Let* $\Gamma^L_{TAG} = \langle \{a, b, c, d, \varepsilon, S_t, S_t^0\}, \{S, S', \overline{S}_1, \overline{S}_2, \overline{a}, \overline{b}, \overline{c}, \overline{d}\}, S', \mathsf{P}\rangle$ *with* P *given as follows*

$$\begin{aligned}
S' &\longrightarrow c_{(1,0)}(S, \varepsilon)\\
S &\longrightarrow c_{(1,1)}(\overline{S}_1, c_{(1,1)}(S, c_{(1,1)}(\overline{S}_2, \pi_1^1)))\\
S &\longrightarrow c_{(1,1)}(S_t^0, \pi_1^1)\\
\overline{S}_1 &\longrightarrow c_{(3,1)}(S_t, a, \pi_1^1, d)\\
\overline{S}_2 &\longrightarrow c_{(3,1)}(S_t, b, \pi_1^1, c)
\end{aligned}$$

Note that we now have only null-ary operatives but extra composition and projection symbols.

We parallel the derivation given in Figure 2 with the new grammar as given in Figure 3. Note that nonterminals are now simply replaced by entire subtrees and no extra insertions take place.

4 Coding the lifted Structures

In the previous section we have shown how to code the non-CF structures with regular tree grammars via LIFTing. In the next section, we will show how to code RTGs themselves with finite-state tree automata (FSTAs) and MSO logic.

4.1 Tree Automata

Since Γ^L_{TAG} in (3.2) generates a regular set of trees, we can construct a tree automaton $\mathfrak{A}_{\Gamma^L_{TAG}} = \langle Q, \Sigma, \delta, q_0, Q_f\rangle$ to recognize this set.

Recall that a (deterministic) *bottom-up tree automaton* (FSTA) $\mathfrak{A}$ 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 $h_\delta(\sigma(t_1, \ldots, t_n)) = \delta(h_\delta(t_1), \ldots, h_\delta(t_n), \sigma)$, $t_i \in T_\Sigma, 1 \le i \le n, \sigma \in \Sigma_n$. 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 $T(\mathfrak{A}) = \{t \mid h_\delta(t) \in F\}$.

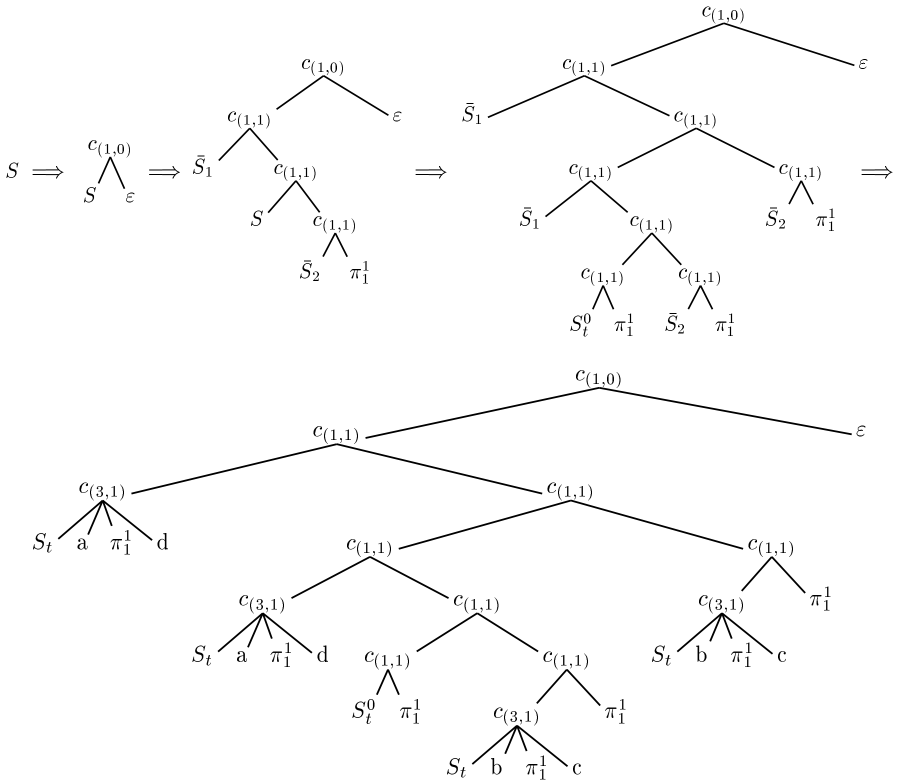


Fig. 3. An example derivation of the LIFTed MCFTG Γ^L_{TAG} given in Example 3.2

Intuitively, such a tree automaton starts on the leaves of a tree in an initial state and then moves toward the root by making transitions according to the states "labeling" the daughters and the alphabet symbol labeling the mother.

Construction of a tree automaton from a given lifted context-free tree grammar $\Gamma^L = \langle \Sigma^L, \mathsf{F}^L, S', \mathsf{P}^L \rangle$, i.e., an RTG, is straightforward. Since tree automata recognize only local trees in each transition, we have to use auxiliary transitions for RHSs of LIFTed macro productions consisting of trees of depth greater than one in order to recognize the correct trees incrementally. So, what we are doing is to decompose the RHSs into trees of depth one which then can be recognized by a transition, i.e., in a preliminary step we have to transform Γ^L into a normal form $\Gamma^{NF} = \langle \Sigma^L, \mathsf{F}^{NF}, S', \mathsf{P}^{NF} \rangle$ via the introduction of auxiliary rules and new nonterminals. In our example, the lifted tree grammar is not in the desired normal form, but it is easy to see how to change this. The resulting rules and nonterminals are reflected both in the new transitions and in the states we need. In the following, we assume without loss of generality, that the trees on the RHSs of the LIFTed macro productions are of depth one.

Recall furthermore that, according to the definition above, a tree automaton operates on a ranked alphabet $\Sigma = \langle \Sigma_n \mid n \in \mathbb{N} \rangle$. Therefore, in our case, we use the inoperative symbols of the lifted grammar to construct Σ, but we reduce the explicit many-sorted type information by defining Σ_n as $\{\sigma \in \Sigma^L \mid rank(\sigma) = n\}$. For the set of states Q, we need distinguishable states for each of the terminals, nonterminals and projection symbols appearing in RHSs of the rules, i.e., $Q = \{q_\sigma \mid \sigma \in \Sigma^L_{\varepsilon,s} \cup \mathsf{F}^{NF}\} \cup \{q_0\}$.[7] Furthermore, we need a new initial state q_0. In the automaton, the state which corresponds to the start symbol S' of the grammar becomes the single final state, i.e., $Q_f = \{q_{S'}\}$.

Since our tree automata work bottom up, we have to start the processing at the bottom by having transitions from the new initial state to a new state encoding the fact that we read a particular symbol on the frontier of the tree. Accordingly, together with the transitions encoding the productions, we have to construct two kinds of transitions in δ:

- transitions from the initial state on all subtrees reading a terminal σ, i.e., elements of all the Σ_i from Γ, to the corresponding state; i.e., $q_0 \times \sigma \rightarrow q_\sigma$;
- transitions recognizing the internal structure of the local trees appearing in RHSs, i.e., from the states corresponding to the leaves of a tree on a RHS to the nonterminal D of the left hand side, i.e., for each lifted tree grammar production of depth one $D \longrightarrow c(d_1, \ldots, d_n)$ we have to construct a transition in the automaton as follows: $q_{d_1} \times \cdots \times q_{d_n} \times c \rightarrow q_D$.

4.2 MSO Logic

Alternatively, we can also code RTGs with MSO logic. The standard way of doing this requires that we first construct the corresponding tree automaton.

In Thomas [33] tree automata are converted to formulas in MSO logic by basically encoding their behaviour. Under the assumption that $Q = \{0, \ldots, m\}$ with $q_0 = 0$, the (closed) Σ^1_1-formula $\varphi_{\mathfrak{A}_{\Gamma^L_{TAG}}}$ given there adapted to our signature and for maximally 4-ary tree automata reads as given below.[8]

Intuitively, the sets X_i label the tree where the automaton assumes state i. The first two lines of the formula say that we cannot have a node which is in two states and that X_0 is our "initial" set; the third one licenses the distribution of the sets according to the transitions and the last one says that we need a root node which is in a "final" set.

$$\begin{aligned}
\varphi_{\mathfrak{A}_{\Gamma^L_{TAG}}} \stackrel{def}{\Longleftrightarrow} (\exists X_0, \ldots, X_m)[\bigwedge_{i \neq j} (\neg\exists y)[y \in X_i \wedge y \in X_j] \wedge \\
(\forall x)[\mathsf{leaf}(x) \rightarrow x \in X_0] \\
\bigwedge_{1 \leq l \leq 4} (\forall x_1, \ldots, x_l, y)[\bigvee_{\substack{(i_1,\ldots,i_l,\sigma,j) \in \delta \\ 1 \leq k \leq l}} x_k \in X_{i_k} \wedge y \triangleleft x_k \wedge y \in X_j \wedge y \in P_\sigma] \\
\bigvee_{i \in Q_f} (\exists x \forall y)[x \triangleleft^* y \wedge x \in X_i]
\end{aligned}$$

[7] We do not need states for the composition symbols since each composition corresponds to a nonterminal due to the normal form.

[8] P_a stands for the predicate labeling a node with the symbol a and $\mathsf{leaf}(x) \stackrel{def}{\Longleftrightarrow} (\neg\exists y)[x \triangleleft y]$.

In Kolb et al. [19] we also propose a way of directly coding the behaviour of the RTG with logical formulas. Since this does not contribute any new insights the interested reader is referred to the earlier paper for further information.

5 Reconstructing lifted (M)CFTGs

Unfortunately, the terminal trees in Figure 3 generated/recognized by the grammar Γ^L_{TAG} given in the Example 3.2, don't seem to have much in common with the structures linguists want to talk about, i.e., the ones in Figure 2.

However, in some sense to be made operational, the LIFTed structures contain the intended structures. As mentioned before, there is a mapping h from these explicit structures onto structures interpreting the compositions (the c's) and the projections (the π's) the way the names we have given them suggest, *viz.* as compositions and projections, respectively, which are, in fact, exactly the intended structures.

On the denotational side, we can implement the mapping h with an MSO definable tree transduction (as defined in Courcelle [5]) and on the operational side with both tree-walking automata (FSTWA, see [2]) and Macro Tree Transducer (MTT, see [8]) to transform the LIFTed structures into the intended ones. In this paper, we will focus on the logical transduction.

Let us restate our goal then: Rogers [29] has shown the suitability of an MSO description language $L^2_{K,P}$ 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 built upon finite-state tree-walking automata (FSTWA) thereby implementing the unique homomorphism mapping the terms into elements of the corresponding context-free tree language, i.e., the trees linguists want to talk about.

Put differently, it should be possible to define a set of relations $R^I = \{\blacktriangleleft, \blacktriangleleft^+, \blacktriangleleft^*$ *(dominance)*, *c-command*, $\lessdot$ *(precedence)*, $\ldots\}$ holding between the nodes $n \in N^L$ of the explicit or LIFTed tree T^L which carry a *"linguistic"* label L in such a way, that when interpreting $\blacktriangleleft^* \in R^I$ as a tree order on the set of *"linguistic"* nodes and $\lessdot \in R^I$ as the precedence relation on the resulting structure, we have a "new" description language on the intended structures.

As mentioned before, we will use an MSO definable tree transduction to transform the LIFTed structures into the intended ones. The core of this transduction will be the definition of the new relations via tree-walking automata.

To do so, it is helpful to note a few general facts (illustrated in Figure 4 with another rendering of the last tree of the derivation given in Figure 3, [17]):

(i) Our trees—that contain no substitutable elements from F_0 any more—feature three families of labels: the "linguistic" symbols, i.e., the lifted in-operatives of the underlying macro-grammar, $\mathsf{L} = \text{LIFT}(\bigcup_{n\geq 0} \Sigma_n)$; the

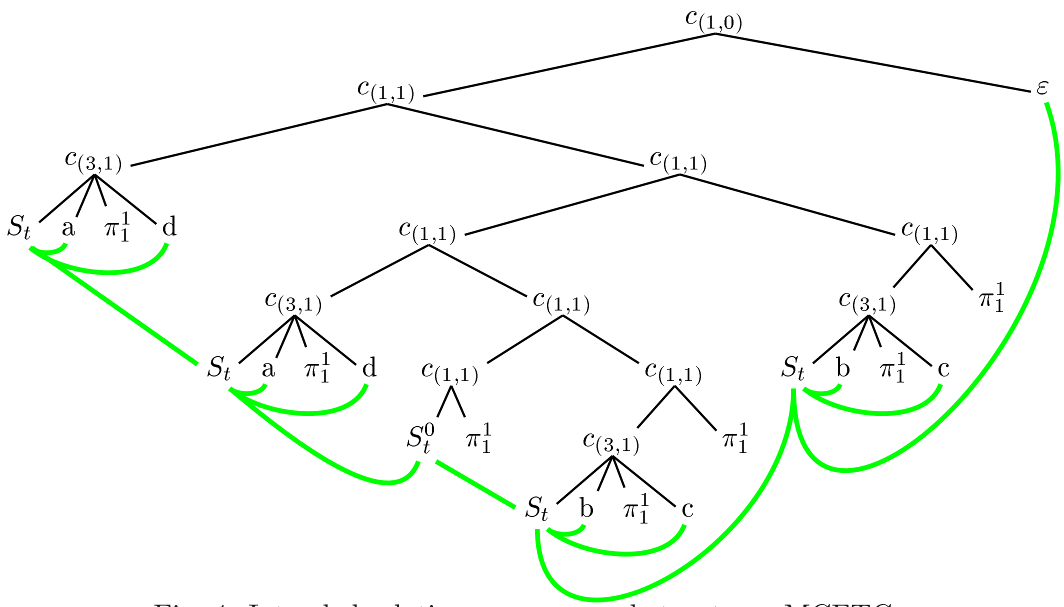


Fig. 4. Intended relations on a LIFTed structure: MCFTGs

"composition" symbols $\mathsf{C} = \{c_{(n,k)}\}, n, k \geq 0$; and the "projection" symbols Π.

(ii) All non-terminal nodes in T^L are labeled by some $c_{(n,k)} \in \mathsf{C}$. This is due to the fact that the "composition" symbols are the only non-terminals of a LIFTed grammar. No terminal node is labeled by some $c_{(n,k)}$.

(iii) The terminal nodes in T^L are either labeled by some "linguistic" symbol or by some "projection" symbol $\pi_i \in \Pi$.

(iv) Any "linguistic" node dominating anything in the intended tree is on some left branch in T^L, i.e., it is the left-most daughter of some $c_{(n,k)} \in \mathsf{C}$. This lies in the nature of composition: $c_{(n,k)}(x_0, x_1, \ldots, x_n)$ evaluates to $x_0(x_1, \ldots, x_n)$.

(v) For any node p labeled with some "projection" symbol $\pi_i \in \Pi$ in T^L there is a unique node μ (labeled with some $c_{(n,k)} \in \mathsf{C}$ by (ii.)) which properly dominates p and whose i-th sister will eventually evaluate to the value of π. Moreover, μ will be the first node properly dominating p which is on a left branch. This crucial fact is arrived at by an easy induction on the construction of Γ^L from Γ.

By (iv.) it is not hard to find possible dominants in any T^L. It is the problem of determining the actual "filler" of a candidate-dominee which is at the origin of the complexity of the definition of $\blacktriangleleft$. There are three cases to account for:

(vi) If the node considered carries a "linguistic" label, it evaluates to itself;

(vii) if it has a "composition" label $c_{(n,k)}$, it evaluates to whatever its function symbol—by (iv.) its leftmost daughter—evaluates to;

(viii) if it carries a "projection" label π_i, it evaluates to whatever the node it "points to"—by (v.) the i^{th} sister of the first C-node on a left branch dominating it—evaluates to.

Note that cases (vii.) and (viii.) are inherently recursive such that a simple MSO definition will not do.

5.1 Reconstruction with FSTWAs

In general, recursive definitions in MSO may lead to undecidability and are therefore properly extending the expressive power of our logic. Fortunately, there are certain techniques to ensure that some relation R which would most naturally be defined recursively has a valid MSO definition. Still, special care has to be taken to establish that the relations defined are well-behaved in this respect. In our case this caveat applies to $\blacktriangleleft$ as well as to its reflexive transitive closure $\blacktriangleleft^*$.

Following Bloem and Engelfriet [2], we will use a *(basic) tree-walking automaton with node-label tests* to specify the *intended* (immediate) dominance relation on LIFTed trees (indicated in Figure 4 with solid green lines), thus showing in passing that it is a regular tree node relation which is indeed MSO definable.

Intuitively, those automata—which are a variation of the tree-walking automata introduced in Aho and Ullman [1]—make transitions from nodes in a tree to other nodes along its branches. Each single transition can either test a label of a node, move up in the tree or down to a specific daughter.

A *tree-walking automaton (with tests)* over some ranked alphabet 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".

For any tree t, such an automaton $\mathfrak{A}$ computes a node relation

$$R_t(\mathfrak{A}) = \{(x, y) \mid (x, q_i) \stackrel{*}{\Longrightarrow} (y, q_f) \text{ for some } q_i \in I \text{ and } q_f \in F\}$$

where for all states $q_i, q_j \in Q$ and nodes x, y in t $(x, q_i) \Longrightarrow (y, q_j)$ iff $\exists d \in \Delta :$ $(q_i, d, q_j) \in \delta$ 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 come from the initial to the final node in a path.

If all the tests $\varphi(x)$ of $\mathfrak{A}$ are definable in MSO logic, $\mathfrak{A}$ specifies a *regular tree-node relation*. Bloem and Engelfriet [2], who should be consulted for details, prove that any regular tree-node relation is itself MSO definable and provide a general translation of $\mathfrak{A}$ into an MSO formula. We, however, will not exploit the full power of the definition: a *basic* tree-walking automaton restricting φ to simple tests of node labels (which are trivially MSO definable via the membership relation) is sufficient for our purposes.

We define $\mathfrak{A}_{\blacktriangleleft} = (Q, \Delta, \delta, I, F)$ with states Q, directives Δ, transitions δ and the initial and final states $I \subseteq Q$ and $F \subseteq Q$, respectively, as given graphically in Figure 5. Construction of the automaton is based upon the careful analysis of the trees involved as given above. Basically, we have to "undo" the LIFTing process by associating the old function symbols, i.e., the intended interior nodes, via the composition symbols with their respective daughters. So it basically implements the facts we presented in (vi.) to (viii.) above.

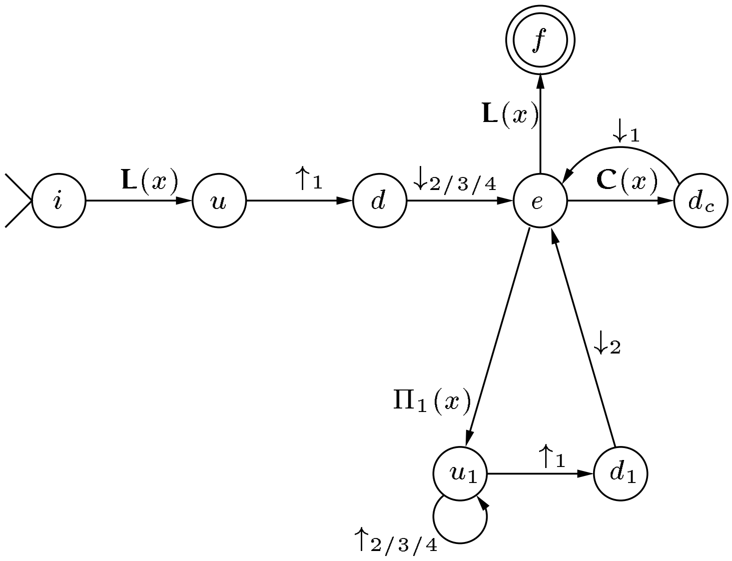


Fig. 5. The FSTWA for dominance on intended structures: MCFTGs

The automaton works as follows: First of all, if it can read a "linguistic" label on the node it started in and if that node is the first (leftmost) daughter, it goes to one of its sisters. Then depending on the case it is in, it "reads" a linguistic label and halts, or reads a composition symbol and finds the leftmost daughter, or it reads a projection symbol and finds the corresponding node by walking upward until it is on a leftmost daughter and then goes on to finding the second sister. This automaton is universal for MCFTGs.[9]

However, there is another interpretation of such a tree-walking automaton. Viewed as an ordinary FSA over the alphabet of directives Δ, $\mathfrak{A}$ recognizes a regular (string-) language, the *walking language* W. In our case it is the walking-language $W_{\blacktriangleleft} = \mathsf{L}(x) \cdot \uparrow_1 \cdot (\downarrow_2 \cup \downarrow_3 \cup \downarrow_4) \cdot (\bigcup_{1 \leq i \leq k} W_{\Pi_i} \cup W_{\mathrm{C}})^* \cdot \mathsf{L}(x)$

[9] The automaton is also universal for CFTGs in the sense that the only variable part it contains is the number of projection functions we have to deal with, i.e., the number of the needed "triangles" in the lower part of the automaton.

with

$$\begin{aligned} W_{\mathsf{C}} &= \mathsf{C}(x) \cdot \downarrow_1 \\ W_{\Pi_i} &= \Pi_i(x) \ \cdot \ (\uparrow_2 \cup \uparrow_3 \cup \uparrow_4)^* \cdot \uparrow_1 \cdot \downarrow_{i+1} \end{aligned}$$

which is finally translated into an MSO-formula $\mathsf{trans}_{W_\triangleleft}(x, y)$. For an example of such a translation, see [27].

For the case of the recursion inherent in reflexive dominance a standard solution exists in MSO logic on finite trees. It is a well-known fact [4] 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:

$$R\text{-}\mathsf{closed}(X) \overset{def}{\Longleftrightarrow} (\forall x, y)[x \in X \wedge R(x, y) \rightarrow y \in X]$$

Now, for any node n, the intersection of all such sets which contain n is exactly the set of m, such that $R^*(n, m)$. Since we are dealing with the (necessarily finite) trees generated by a context-free grammar, this construction can be safely exploited for our purposes;

Similarly to the tree-walking automaton for dominance, we also have to construct a tree-walking automaton for the intended precedence relation. Since the RTGs in all the examples we have given are *linear* RTGs, i.e, they do not use a variable more than once on any RHS, we can present the definition for precedence in two steps as follows.

The first step is defined with an FSTWA which encodes the *immediate* precedence relations. Then we define precedence following the same reasoning as before, namely as *proper* precedence, based upon the definition of immediate precedence.

Looking at the example in Figure 4, what can we observe concerning precedence? The linguistically labeled terminals seem to appear in the right order such that precedence reduces to sisterhood. But what happens to the intended interior nodes? And how do we have to treat the projection nodes? The answer appears to be simple: the nonterminals have to be inserted for the "right" projection symbols and then precedence is reducible to sisterhood. But finding the correct projection symbol for a filler is non-trivial.

The FSTWA given in Figure 6 basically consists of two parts: the first part deals with those cases where the nodes which are labeled with a linguistic label are already sisters, i.e., the intended terminal nodes, or have a node labeled with a projection node as sister, i.e., they precede a nonterminal. Between them we simply have an immediate precedence relation which, in the case of a projection node, has to be found via a further traversal of the tree. In Figure 6 those nodes are identified by going up a non-leftmost branch, and then by descending to the non-leftmost sisters. If the node found bears a linguistic label, we are done. If it bears a projection symbol, we proceed analogously to the definitions for dominance, i.e., go up until we find a leftmost node and

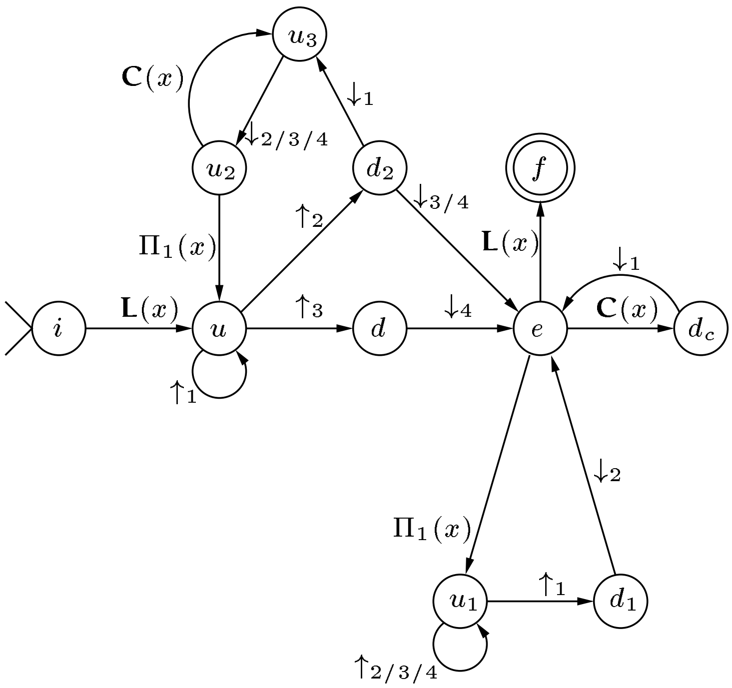


Fig. 6. The FSTWA for precedence on intended structures: MCFTGs

then go to down the right branch.

The situation is complicated by the fact that every node with a linguistic label at the same time is a potential filler for a projection node.[10] Basically, we have to reverse the process which finds a filler for a projection node. But since a projection line can consist of several projection nodes, how do we know when to stop? The FSTWA ensures that the relevant projection nodes for a filler are found by recursively looking up and then down a tree. The nodes bearing a linguistic label which are on a leftmost branch have to look upwards on leftmost branches until they find the first right branch. At this point they have to descend again into the subtree to find the projection node. This is achieved by repeatedly going down non-leftmost branches as long as there are composition nodes on the way and testing for a projection node. If we find one, we can start computing its sisters. If it doesn't have any, it starts the entire process recursively. The resulting relation will be indicated by $\lhd_{imm}$ and the corresponding formula generated by the procedure outlined previously $\mathsf{trans}_{W_{\lhd_{imm}}}$.

We are now in a position to complete the definition of precedence by using the definition of immediate precedence. As usual, two nodes stand in the precedence relation if they either stand in the immediate precedence relation or if they are dominated by nodes which stand in the immediate precedence

[10] In this particular example, where we have only one projection node, the only potential fillers will either be a second daughter, or the leftmost daughter of a composition node which is a right daughter.

relation:

$$x \lhd y \stackrel{def}{\Longleftrightarrow} (\exists u, v)[u \blacktriangleleft^* x \wedge v \blacktriangleleft^* y \wedge \mathsf{trans}_{W_{\lhd_{imm}}}(u, v)]$$

5.2 Reconstruction with MSO Transductions

Now we can turn to the definition of the transduction via MSO logic. Since we did the necessary preparatory work with the presentation of the needed tree-walking automata, we are almost done.

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 $r_{\mathcal{R}} \subseteq D_{\mathcal{R}}^{\rho(r)}$. We can code trees as relational structures by taking a tree domain as the domain $D_{\mathcal{R}_{w,A}}$ of the structure and defining *suc* as the corresponding tree order.

The classical technique of interpreting a relational structure 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.

Definition 5.1 [MSO transduction] Let R and Q be two finite sets of ranked relation symbols. 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 $(\varphi, \psi, (\theta_q)_{q \in Q})$ consisting of an MSO formula φ defining the domain of the transduction in $\mathcal{R}$, an MSO formula ψ defining the resulting domain of $\mathcal{Q}$, and a family of MSO formulas θ_q defining the new relations Q using only definable formulas from the "old" structure $\mathcal{R}$, i.e., for α a variable assignment,

$$D_{\mathcal{Q}} = \{d \in D_{\mathcal{R}} \mid (\mathcal{R}, d) \models \psi[\alpha]\}$$

and for each $q \in Q$

$$q_{\mathcal{Q}} = \{(d_1, \ldots, d_n) \in D_{\mathcal{Q}}^n \mid (\mathcal{R}, d_1, \ldots, d_n) \models \theta_q[\alpha]\} \text{ where } n = \rho(q)$$

Note that the transduction is only defined if φ holds.

Using the defined formulas for $\blacktriangleleft$ and $\lhd$, the specific MSO transduction we need to transform the LIFTed structures into the intended ones simply looks as follows:

$$(\varphi, \psi, (\theta_q)_{q \in Q})$$

$$Q = \{\blacktriangleleft, \blacktriangleleft^*, \blacktriangleleft^+, \lessdot, \dots\}$$

$$\begin{aligned}
\varphi &\equiv \varphi_{\mathfrak{A}_{\Gamma^L_{TAG}}} \\
\psi &\equiv \mathsf{L}(x) \\
\theta_{\blacktriangleleft}(x,y) &\equiv \mathsf{trans}_{W_{\blacktriangleleft}}(x,y) \\
\theta_{\blacktriangleleft^*}(x,y) &\equiv (\forall X)[\blacktriangleleft\text{-}\mathsf{closed}(X) \wedge x \in X \rightarrow y \in X] \\
\theta_{\blacktriangleleft^+}(x,y) &\equiv x \blacktriangleleft^* y \vee x \not\approx y \\
\theta_{\lessdot}(x,y) &\equiv x \lessdot y \\
\theta_{\text{labels}} &\equiv \text{taken over from } R
\end{aligned}$$

As desired, the domain of the transduction is characterized by the MSO formula for the LIFTed trees (see Section 4). The domain, i.e., the set of nodes, of the intended tree is characterized by the formula ψ which identifies the nodes via the "linguistic" labels. Building on it, we define the other primitives of our description language analogous to $L^2_{K,P}$ with the given FSTWAs.

Note that while standard "linguistic" relations like *c-command* or *government* would be defined in terms of *dominance*, our approach allows the alternative route of taking, in the spirit of Frank and Vijay-Shanker [11], *c-command* as the primitive relation of linguistic structure by defining, in a similar fashion, an FSTWA which computes the intended *c-command* relation directly, without recourse to dominance.

6 Conclusion

All the constructions that have been cited as evidence for the need of assuming context-sensitive grammatical devices for the description of natural languages seem to be amenable to an analysis within the framework of context-free tree grammars or multiple context-free grammars [18,20,23,26,27]. Based on the Mezei-Wright result [22] according to which structural accounts of the context-free tree level can be lifted to the regular tree level where composition and projection occur as explicit node labels, the paper has focused on a fine-grained analysis of the process connecting the initial level of explicit trees with the intended level of context-free or macro trees. In accordance with the three types of classical approaches in formal language theory we have provided logical, grammatical and automata-theoretic characterizations of the homomorphism relating the initial tree algebra of terms to the substitution algebra of macro terms. Along the way first steps were taken in the direction of "reverse" linguistics. For the contemporary natural language formalism of

Tree Adjoining Grammar it has been shown how to accommodate the main ideas within the algebraic framework outlined above.

Comparing this statement of the result of the paper with the characterization of context-free graph languages by Engelfriet and van Oostrom [9], we want to stress the point that our logical description of (M)CFTG languages does not provide a characterization of this language family in the technical understanding of an equivalence between MCFTG languages and languages defined by a regular tree language/closed MSO formula and a macro tree transducer/MSO transduction. For a recent result on the equivalence between regular tree languages followed by an MSO definable tree transduction and the tree languages generated by context-free graph grammars see Engelfriet and Maneth [8].

References

[1] Aho, A. V. and J. D. Ullman, *Translations on a context-free grammar*, Information and Control **19** (1971), pp. 439–475.

[2] Bloem, R. and J. Engelfriet, *Characterization of properties and relations defined in Monadic Second Order logic on the nodes of trees*, Technical Report 97-03, Dept. of Computer Science, Leiden University (1997).

[3] Büchi, J. R., *Weak second-order arithmetic and finite automata*, Z. Math. Logik Grundlag. Math. **6** (1960), pp. 66–92.

[4] Courcelle, B., *Graph rewriting: An algebraic and logic approach*, in: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Handbook of Theoretical Computer Science **B**, Elsevier, 1990 pp. 193–242.

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

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

[7] Elgot, C. C., *Decision problems of finite automata design and related arithmetics*, Trans. Amer. Math. Soc. **98** (1961), pp. 21–51.

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

[9] Engelfriet, J. and V. van Oostrom, *Regular description of context-free graph languages*, Journal Comp. & Syst. Sci. **53** (1996), pp. 556–574.

[10] Fischer, M. J., *Grammars with macro-like productions*, in: *Proceedings of the 9th Annual Symposium on Switching and Automata Theory*, IEEE, 1968, pp. 131–142.

[11] Frank, R. and K. Vijay-Shanker, *Primitive c-command* (1998), ms., Johns Hopkins University & Universtiy of Delaware.

[12] Fujiyoshi and Kasai, *Spinal-formed context-free tree grammars*, MST: Mathematical Systems Theory **33** (2000).

[13] Joshi, A. K., *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural description*, in: D. Dowty, L. Karttunen and A. Zwicky, editors, *Natural Language Parsing*, Cambridge University Press, Cambridge, UK, 1985 pp. 206–250.

[14] Joshi, A. K., *An introduction to tree adjoining grammars*, in: Manaster-Ramer, editor, *Mathematics of Language*, John Benjamins, Amsterdam, 1987 .

[15] Joshi, A. K., L. S. Levy and M. Takahashi, *Tree adjunct grammars*, Journal of Computer and System Sciences **10** (1975), pp. 136–63.

[16] Joshi, A. K. and Y. Schabes, *Tree adjoining grammars*, in: G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, Handbook of Formal Languages **3: Beyond Words**, Springer, Berlin, 1997 pp. 69–123.

[17] Kolb, H.-P., *Macros for minimalism? Towards weak descriptions of strong structures*, in: H.-P. Kolb and U. Mönnich, editors, *Mathematics of Syntactic Structure: Trees and Their Logics*, Studies in Generative Grammar **44**, Mouton-de Gruyter, 1999 .

[18] Kolb, H.-P., J. Michaelis, U. Mönnich and F. Morawietz, *An operational and denotational approach to non-context-freeness* (2000), to appear in: *Theoretical Computer Science*, Elesevier Science. Draft available under `http://tcl.sfs.uni-tuebingen.de/~frank/`.

[19] Kolb, H.-P., U. Mönnich and F. Morawietz, *Logical description of cross-serial dependencies*, in: *Proceedings of the Workshop Finite Model Theory and its Applications at FloC '99*, Trento, Italy, 1999.

[20] Kolb, H.-P., U. Mönnich and F. Morawietz, *Descriptions of cross-serial dependencies*, Grammars **3** (2000), pp. 189–216.

[21] Maibaum, T. S. E., *A generalized approach to formal languages*, J. Comput. System Sci. **88** (1974), pp. 409–439.

[22] Mezei, J. and J. B. Wright, *Algebraic automata and contextfree sets*, Information and Control **11** (1967), pp. 3–29.

[23] Michaelis, J., U. Mönnich and F. Morawietz, *On minimalist attribute grammars and macro tree transducers* (2000), to appear in: *Linguistic Form and its Computation*, Ch. Rohrer, A. Rossdeutscher, H. Kamp (eds), CSLI. Draft available under `http://tcl.sfs.uni-tuebingen.de/~frank/`.

[24] Mönnich, U., *Adjunction as substitution*, in: G.-J. M. Kruijff, G. Morill and R. Oehrle, editors, *Formal Grammar '97*, Aix-en-Provence, 1997, pp. 169–178.

[25] Mönnich, U., *On cloning contextfreeness*, in: H.-P. Kolb and U. Mönnich, editors, *The Mathematics of Syntactic Structure*, number 44 in Studies in Generative Grammar, Mouton de Gruyter, 1999 pp. 195–229.

[26] Mönnich, U., *Regular Description of HPSG* (2001), ms, Universität Tübingen.

[27] Morawietz, F., "Two-Step Approches to Natural Language Formalisms," Ph.D. thesis, Universität Tübingen (2001).

[28] Pullum, G. K. and B. C. Scholz, *On the Distinction between Model-Theoretic and Generative-Enumerative syntactic Frameworks*, in: P. de Groote, G. Morill and C. Retoré, editors, *Logical Aspects of Computational Linguistics* (2001), pp. 17–43, LNAI 2099.

[29] Rogers, J., "A Descriptive Approach to Language-Theoretic Complexity," Studies in Logic, Language, and Information, CSLI Publications and FoLLI, 1998.

[30] Rogers, J., *A descriptive characterization of tree-adjoining languages*, in: *Proceedings of COLING-ACL 98*, 1998.

[31] Rounds, W. C., *Tree-oriented proofs of some theorems on context-free and indexed languages*, in: *Proceedings of the 2nd Annual ACM Symposium on Theory of Computing*, 1970, pp. 109–116.

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

[33] Thomas, W., *Automata on infinite objects*, in: J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Elsevier Science Publishers B. V., 1990 pp. 133–191.

[34] Vijay-Shanker, K. and D. J. Weir, *The equivalence of four extensions of context-free grammars*, Mathematical Systems Theory **27** (1994), pp. 511–546.