A Two-Step Approach to Non-Contextfreeness

Jens Michaelis, Uwe Mönnich and Frank Morawietz

Seminar für Sprachwissenschaft
Universität Tübingen,
Wilhelmstr. 113,
72074 Tübingen
{michael,um,frank}@sfs.nphil.uni-tuebingen.de

Abstract. Extending 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), we present in this paper a translation from MCFGs into regular tree grammars (which are characterizable by both tree automata and a corresponding formula in monadic second-order (MSO) logic) by viewing them as terms in a free Lawvere theory. These are then unpacked with an MSO transduction (which can be implemented as a Macro tree Transducer) based upon tree-walking automata, thereby giving an operational and 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.

The weak equivalence of a rich class of (mildly context-sensitive) generative formalisms has been proven over the last couple of years, e.g., the following language families 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 the 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, for example, be found in Engelfriet, 1997; Rambow and Satta, 1999; Weir, 1992; Michaelis, 1999; Kolb et al., 1999a,b).

The main result of this paper is a description of non context-free phenomena equivalently in terms of (a) regular tree languages (to express the recursive properties) and regular string languages (to establish the intended linguistic relations) and (b) transductions definable in monadic second-order (MSO) logic which are defined on a domain of finite trees that is characterized as the model set of a closed MSO formula. In fact, we shall not deal directly with the two types of regular language families and MSO definable structures, but shall instead consider implementation in the form of finite-state (tree) devices to emphasize the effectivity of our approach.

In the present context the algebraic, logical and regular perspective not only adds another characterization of mildly context-sensitive tree languages to the already long list of equivalences with restricted production systems, but it also makes available the whole body of techniques that have been developed in the tradition of algebraic language theory, logic and automata theory.

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 by providing translation procedures that transform logical specifications into finite automata equivalent to the language classes and vice versa. To be a little more specific, Büchi (1960) and Elgot (1961) have shown that regular string languages represented through finite (string) automata can be expressed by sentences in the weak monadic second-order logic with one successor. For tree languages an analogous result is well known to the effect that a tree language is definable in weak monadic second-order logic with multiple successors if and only if it is recognizable by a finite tree automaton (Doner, 1970; Thatcher and Wright, 1968).

It is these earlier characterizations that provide the formal foundations 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.

The logical approach to the specification of language classes has a lot of advantageous properties that have paved the way to its application to linguistic issues. First, the equivalence between automata theoretic operational and logic oriented declarative formalisms leads to a lot of closure properties of the defined language classes that follow immediately from the closure of the specification logics with respect to the traditional operations like negation, conjunction, alternation and (universal and existential) quantification. Second, the transition from strings to finite model-theoretic structures of arbitrary signatures requires no extra conceptual or technical ideas in the logical framework whereas in formal language theory the step from string to tree languages and the concomitant distinction between weak and strong generative capacity constitutes a significant extension of the research agenda. Third, since the logical approach does not depend on an operational process which, starting from some given objects, generates its space of interpretation, but refers directly to an assumed universe of structures, its statements can be phrased in terms of linguistically significant notions that enter into universal principles and language-particular constraints. Finally, those logical languages that capture complexity classes indicate lower bounds on the computing resources a system has to make available that wants to use those structures that fall within the classes correlated with the corresponding logical language.

The latest ideas in Generative Grammar in the Chomsky tradition, the so called *Minimalist Program* (Chomsky, 1995), allow the specification of non context-free languages (Stabler, 1997). Those are necessary in the formal analysis of natural languages for example to account for the phenomena of cross-serial dependencies. E.g., in a dialect of Swiss German, verbs and their objects exhibit dependencies by overt case-marking. Since the blocks of verbs and the corresponding blocks of objects internally occur in the same order, their dependencies have to cross each other. Consider the following example as an illustration:

(1) a) *(...weil) der Karl die Maria dem Peter den Hans schwimmen lehren helfen liess*

(German: Palindrome language – CF)

b) *(...omdat) Karel Marie Piet Jan liett helpen leren zwemmen*

(Dutch: $a^n b^n$ – CF)

c) *(...wil) de Karl d'Maria em Peter de Hans lönd hälffe lärne schwüme*

(Swiss German: ww – Non-CF)

(...because) Charles Mary Peter John lets help to teach to swim

"...because Charles lets Mary help Peter to teach John to swim"

It has been shown in Kolb et al. (1999a,b) how to cope with these phenomena starting from context-free tree grammars with a similar two-step approach. But the direct connection to the Minimalist Grammars (MG) used by todays generative grammarians is not immediately apparent.

Therefore, extending the work by Michaelis (1999) which shows how to encode an arbitrary 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 which are a weakly equivalent extension of LCFRSs, into regular tree grammars/tree automata. These are then unpacked with a monadic second-order (MSO) transduction which can be implemented with an appropriately extended macro tree transducer (Engelfriet and Maneth, 1999), thereby giving both an operational and a logical description of the tree sets involved.

The basic idea underlying this sequential application of two finite-state devices or two logical specification steps respectively can be found in the signature free treatment of universal algebra that was developed in the early 60s. According to this treatment, the most important rôle in an algebra is played by the set of all operations definable from the primitive ones by composition and not by the primitive operations themselves. These defined operations then constitute the carrier of a (multi-sorted) algebra whose only non-unary operations are provided by suitably sorted instances of composition.

A result similar in spirit to the one established in this paper can be found in Engelfriet and van Oostrom (1996). They show that context-free graph languages have a regular path description. Given the context of graph theory the authors use for their first step a regular family of derivation trees and formalize the effect of connection instructions in terms of suitable node labels in the derivation trees. Similarly, in Courcelle (1992), a certain type of context-free graph languages is characterized as an equational subset of an algebra over a binary glueing operation and a family of definable unary operations.

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 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. In that sense, the translation from MCFGs to regular tree grammars 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.

2 Translating MCFGs to Regular Tree Grammars

The core idea in Michaelis (1999) is that for the set of trees appearing as intermediate steps in converging derivations corresponding to a given MG 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 the corresponding MCFG represents such an equivalence class, i.e., an infinite set of trees. We take these resulting grammars as our starting point.

We are going to show how to code the grammar rules of an MCFG into rules of a regular tree grammar via lifting by viewing them as terms in a free Lawvere theory. Since this coding makes the projections, 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 its domain – to transform them into the intended trees. We first present some basic definitions before we proceed with the actual translation.

2.1 Basic Definitions

Let S be a set of sorts. A *many-sorted signature* Σ is an indexed family $\langle \Sigma_{w,s} \mid w \in S^*, s \in S\rangle$ of disjoint sets. A symbol in $\Sigma_{w,s}$ is called an operator of type $\langle w, s\rangle$, arity w, sort s and rank $\ell(w)$, where $\ell(w)$ denotes the length of w. In the case of a single-sorted signature we write $\Sigma_{s^n,s}$ as Σ_n. The set of k-ary trees $T(\Sigma, X_k)$ over such a single-sorted signature Σ is built up from a finite set $X_k = \{x_1, \ldots, x_k\}$ of variables using the operators in the usual way: If $\sigma \in \Sigma_n$ and $t_1, \ldots, t_n$ are trees then $\sigma(t_1, \ldots, t_n)$ is a k-ary tree. Note that each $x_i \in X_k$ is a trivial tree.

The operator symbols induce operations on an algebra with the appropriate structure. A Σ-*algebra* $\mathbb{A}$ consists of an S-indexed family of sets $A = \langle A_s\rangle_{s\in S}$ and for each operator $\sigma \in \Sigma_{w,s}$, a function $\sigma_{\mathbb{A}} : A^w \to A^s$ where $A^w = A^{w_1} \times\cdots\times A^{w_n}$

and $w = w_1 \cdots w_n$. The set of k-ary trees $T(\Sigma, X_k)$ can be made into a Σ-algebra $\mathbb{T}$ by specifying the operations as follows. For every $\sigma \in \Sigma_n$ and every $t_1, \ldots, t_n \in T(\Sigma, X_k)$ 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*. Given a set of sorts S, an algebraic theory, as an algebra, is an $S^* \times S^*$-sorted algebra $\mathbb{T}$, whose carriers $\langle T(u,v) \mid u, v \in S^* \rangle$ consist of the morphisms of the theory and whose operations are of the following types $(1 \le i \le n; u_i, v \in S^*; u = u_1 \ldots u_n)$:

projection:	$\pi_i^u \in T(u, u_i)$
composition:	$c_{u_1,\ldots,u_n} \in T(u_1, u_2) \times \cdots \times T(u_{n-1}, u_n) \to T(u_1, u_n)$
target tupling:	$(\quad)_{(v,u)} \in T(v, u_1) \times \cdots \times T(v, u_n) \to T(v, u)$

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

With S being a singleton, the power set $\wp(T(\Sigma, X_k))$ of k-ary trees constitutes the central example of interest for formal language theory. The carriers $\langle \wp(T(k,m)) \mid n, m \in w \rangle$ consist of sets of m-tuples of k-ary trees $\{(t_1, \ldots, t_m)\}$. The operation of composition is defined as substitution of the projection constants and target tupling is just tupling. For reasons of space, we cannot go into more details here. More on Lawvere theories in this context, derived operations and their connection to linguistics can be found in Mönnich (1997; 1998).

Very briefly, a *multiple context-free grammar* (MCFG) can be defined as a five-tuple $\mathcal{G} = \langle N, T, F, P, S \rangle$ of ranked nonterminals N, terminals T, basic morphisms F, start symbol $S \in N$ and productions P, where each element of P has the form $A \longrightarrow f(A_0, \ldots, A_{n-1})$ for $A, A_0, \ldots, A_{n-1} \in N$ and $f \in F$ a function from $(T^*)^k$ to $(T^*)^l$ with arity $k = \Sigma_{i=0}^{n-1} k_i$ (k_i the rank of A_i) and l the rank of A (cf. Seki et al. (1991)). Recall that the basic morphisms are those which use only composition, variables and tupling.

A *tree grammar* Γ is a 5-tuple $\langle \Sigma, \mathbf{F}, S, \mathbf{X}, \mathbf{P} \rangle$, where $\Sigma, \mathbf{F}$ are the ranked alphabets of *inoperatives* and *operatives*, respectively; $S \in \mathbf{F}$ is the starting symbol; $\mathbf{X}$ is a countable set of variables; and $\mathbf{P}$ is a set of productions. Each $p \in \mathbf{P}$ has the form $F(x_i, \cdots, x_n) \longrightarrow t$, with $F \in \mathbf{F}$, $x_i, \cdots, x_n \in \mathbf{X}$, and t a term (tree) over $\Sigma \cup \mathbf{F} \cup \{x_i, \cdots, x_n\}$. An application of a rule $F(x_i, \cdots, x_n) \longrightarrow t$ "rewrites" a tree rooted in F as the tree t with its respective variables substituted by F's daughters. Tree grammars with $\mathbf{F}_n = \emptyset, n \neq 0$, are called *regular*. Since they 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 (Mezei and Wright, 1967).[1]

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 then 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

[1] Appropriate definitions for derivations and the tree languages generated can be found in Kolb et al. (1999b).

example of a tree automaton in this abstract, we will not specify further details. More can be found in Gécseg and Steinby (1984).

We need another type of finite-state automata: *tree-walking automata with MSO tests* (Bloem and Engelfriet, 1997). Intuitively, those 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.

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 a tree-walking automaton $\mathfrak{A}$ computes a node relation $R_t(\mathfrak{A}) = \{(x,y) | (x, q_i) \overset{*}{\Rightarrow} (y, q_f) \text{ for some } q_i \in I \text{ and some } q_f \in F\}$ 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.

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.

The following paragraphs go directly back to Courcelle (1997). Representation of objects with relational structures is a standard way to make 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 $r_{\mathcal{R}} \subseteq D_{\mathcal{R}}^{\rho(r)}$.

With the classical technique of interpretations where one relational structure is defined within another one we can overcome the limitation that, at first glance, there does not seem to be a convenient machine model for tree transformations. Fortunately, one can use logic directly to define the desired transduction. We will use this concept to transform the lifted trees into the intended ones.

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 a formula φ defining the domain of the transduction in $\mathcal{R}$, 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 context-free and regular power respectively 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-contextfree phenomena 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.

The machine model for the entire transduction are Macro Tree Transducer (MTT) with regular look-ahead which are constrained to be only finite-copying (Engelfriet and Maneth, 1999). Since the equivalence proof of MSO definable transductions and MTTs is very complex and indirect via a restricted form of *attribute grammars*, we will not present the resulting actual MTT in this paper and therefore refrain from giving a full definition. In short, an MTT is a generalization of a top-down tree-to-string transducer to trees such that context information can be handled. Each transition of an MTT takes a tree and some parameters and transforms these into a new tree which contains the components of the original tree, the parameters and (possibly) state symbols. The regular look-ahead is implemented using an additional tree automaton which ensures that each of the components of the input tree is indeed a member of the defined regular tree language. The restriction to use only a finite mount of copying applies to both the components of the input tree and the parameters and simply means that each of those is copied only a bounded number of times.

We will give the full MSO definition of the transduction we need built with binary MSO relations defined by tree-walking automata, though. The translation into the corresponding MTT is left to the reader.

2.2 The Translation

Each rule of the MCFG is recursively transformed into a regular tree grammar 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 as elements of a free Lawvere algebra. Consider the following example MCFG $\mathcal{G} = \langle \{S, A\}, \{a_1, a_2, a_3\}, S, P \rangle$, where

$$P = \left\{ \begin{array}{l} S \to g(A) \\ A \to f() \\ A \to h(A) \end{array} \right\} \qquad \begin{array}{r} g(x,y,z) = xyz \\ f() = (a_1, a_2, a_3) \\ h(x,y,z) = (xa_1, ya_2, za_3) \end{array} \tag{2}$$

The language generated by $\mathcal{G}$ is $a_1^n a_2^n a_3^n$. The (Lawvere) arity of S, a_1, a_2 and a_3 is $(0,1)$, of A $(0,3)$ and the one of $\bullet, \pi_1^3, \pi_2^3$ and π_3^3 is $(3,1)$. We leave out this typing information unless it is relevant. The regular tree grammar $\mathcal{G}'$ resulting from the translation has rules (in tree notation) which look as given in Fig. 1.

As one can see, the basic functions have been realized as terms with their respective implicit operations as nonterminals (composition and tupling) or terminals (projection). The easiest case is the non-recursive A rule. The function f just returns a triple of a, b and c. The corresponding tree has a mother node with three unary arguments and returns a triple and the appropriate daughters. The S rule is more complicated with the function g concatenating three (input) strings. The definition of the function can be written explicitly as a Lawvere term as $(c(\bullet, c(\bullet, \pi_1^3, \pi_2^3), \pi_3^3)$.[2] Furthermore, note that the implicit binary concatenation $\bullet$ in g now becomes a constant. The variables are simply replaced

[2] For simplicity and readability we ignore the subscripts of the compositions c.

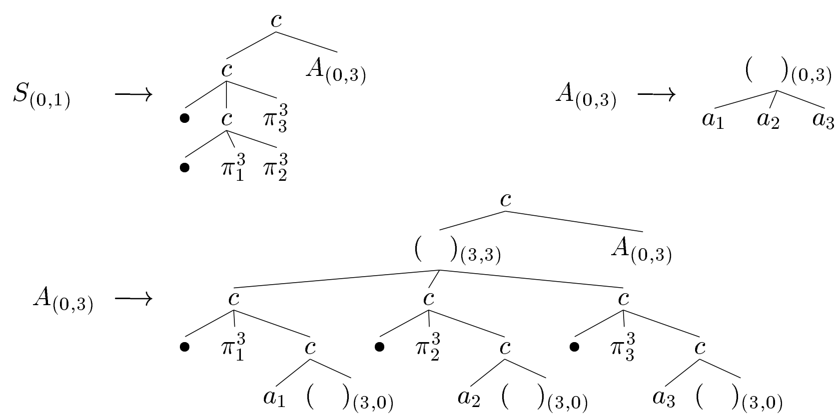

Fig. 1. The translated example grammar

by the projections and concatenated. The resulting term is then applied to the nonterminal A such that we get the RHS displayed in the S rule in Fig. 1.

Since regular tree grammars can only generate recognizable languages, we can characterize them with both MSO logic on trees and tree automata. The tree automaton $\mathfrak{A}_{G'}$ is generated simply by transforming the grammar into a normal form such that each RHS is of depth one by introducing auxiliary nonterminals. 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. Thomas (1990) describes how to transform this tree automaton into an MSO formula $\varphi_{\mathfrak{A}_{G'}}$, basically by encoding its behaviour. A more precise description how to construct both the tree automaton and the corresponding MSO formula can be found in Kolb et al. (2000).

3 Unpacking the Intended Trees

Rogers (1998) has shown the suitability of an MSO description language for linguistics which is based upon the primitive relations of immediate ($\triangleleft$), proper ($\triangleleft^+$) and reflexive ($\triangleleft^*$) dominance and proper precedence ($\prec$). We will show how to define those with an MSO transduction.

As one can see, the trees belonging to the tree language generated by $\mathcal{G}'$ are elements of the free Lawvere term algebra. Therefore there exists a unique homomorphism mapping the terms into elements of the corresponding regular tree language. Even more concretely, we can give an MSO transduction which transforms the given lifted trees into the intended ones. At the core of the transduction is a tree-walking automaton with MSO tests. This automaton defines the binary relation of immediate dominance on the nodes belonging to the intended structures. It is given graphically in Fig. 2.[3]

[3] The technique we are using here and its underlying assumptions have been described in more detail in Kolb et al. (1999b).

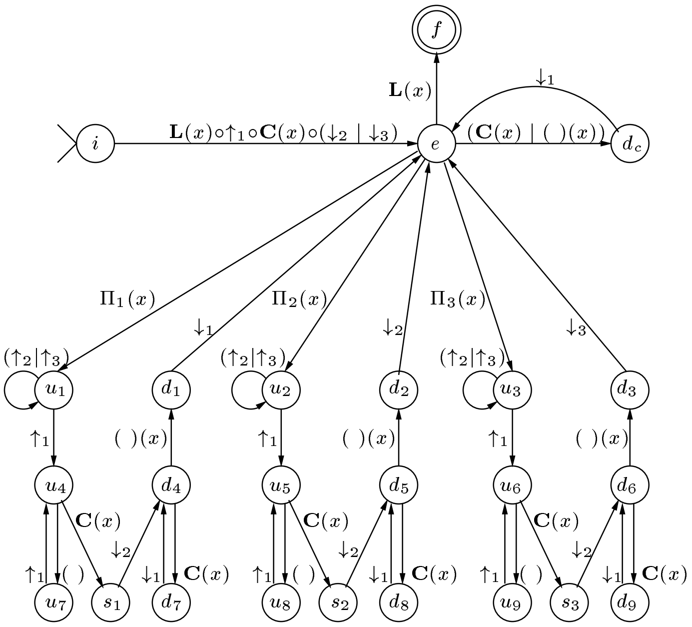

Fig. 2. The tree-walking automaton for dominance

The automaton starts on any node with a "linguistic" label (denoted here by $\mathbf{L}$) which means for the given example $\bullet, a, b, c$. Then it has to go up the first branch, read a composition symbol and then it nondeterministically descends to its sisters (in the example only to the second and third daughter). If they are "linguistic" nodes, the automaton stops. If they are a composition or a tupling symbol, the automaton goes to the leftmost daughter and tries again. If it reads a projection symbol, it finds the appropriate "filler" by going upwards until it is on a leftmost branch which is labeled with a composition symbol. Then it 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. Let us write the resulting dominance relation with $\triangleleft$.

However, there is another interpretation of such an automaton. Viewed as an ordinary finite-state automaton over the alphabet Δ, $\mathfrak{A}$ recognizes a regular (string-) language, the *walking language* W which can be translated recursively into an MSO formula (see Bloem and Engelfriet, 1997). It is this formula trans_W we will use in the definition of the transduction.

Before we can do so, we need one auxiliary definition. To account for the recursion inherent in reflexive dominance, a standard solution exists, at least in *weak* MSO, the monadic second-order logic on *finite* structures. It is a well-known fact (e.g. Courcelle (1990)) 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: $R\text{-}\mathsf{closed}(X) \stackrel{def}{\Longleftrightarrow} (\forall x, y)[x \in X \wedge R(x, y) \rightarrow y \in X]$.

Using the defined formula $\mathsf{trans}_{W_\triangleleft}(x, y)$ for the intended immediate dominance relation $\triangleleft$ the specific MSO transduction $(\varphi, \psi, (\theta_q)_{q\in Q})$ we need to transform the lifted structures into the intended ones looks as follows:

$$
\begin{aligned}
Q &= \{\triangleleft, \triangleleft^*, \triangleleft^+, \prec, \dots\} \\
\varphi &\equiv \varphi_{\mathfrak{A}_{G'}} \\
\psi &\equiv (\exists y)[\mathsf{trans}_{W_\triangleleft}(x, y) \vee \mathsf{trans}_{W_\triangleleft}(y, x)] \\
\theta_\triangleleft(x, y) &\equiv \mathsf{trans}_{W_\triangleleft}(x, y) \\
\theta_{\triangleleft^*}(x, y) &\equiv (\forall X)[\triangleleft\text{-}\mathsf{closed}(X) \wedge x \in X \rightarrow y \in X] \\
\theta_{\triangleleft^+}(x, y) &\equiv x \triangleleft^* y \vee x \not\approx y \\
\theta_\prec(x, y) &\equiv \text{another tree-walking automaton} \\
\theta_{\text{labels}} &\equiv \text{taken over from } R
\end{aligned}
\tag{3}
$$

As desired, the domain of the transduction is characterized by the MSO formula 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.[4]

It is this definition (3) which has to be translated into an attribute grammar which is single use restricted which then in turn is translated into the appropriate MTT (Bloem and Engelfriet, 1998; Engelfriet and Maneth, 1999).

4 Conclusion

Taking the result of Michaelis' translation of MGs as the input we have shown how to define a regular tree grammar 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/macro tree transducers) 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. Citing another result from Engelfriet and Maneth (1999), we also know that our two-step approach of applying an MSO tree translation to a regular tree set results in the tree languages generated by context-free graph grammars.

[4] This is certainly true for lifted structures resulting from *linear* context-free tree grammars. For non-linear ones, we have to take each individual run of the automaton into consideration which complicates matters considerably.

Bibliography

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.

Bloem, R. and Engelfriet, J. (1998). A comparison of tree transductions defined by monadic second order logic and by attribute grammars, *Technical Report 98-02*, Leiden University Technical Report.

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

Chomsky, N. (1995). *The Minimalist Program*, Vol. 28 of *Current Studies in Linguistics*, MIT Press.

Courcelle, B. (1990). Graph rewriting: An algebraic and logic approach, *in* J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science*, Vol. B, Elsevier, pp. 193–242.

Courcelle, B. (1992). The monadic seond-order logics of graphs VII: Graphs as relational structures, *TCS* **102**: 3–33.

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

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

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

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

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

Engelfriet, J. and van Oostrom, V. (1996). Regular description of context-free graph languages, *Journal Comp. & Syst. Sci.* **53**(3): 556–574.

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

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

Kolb, H.-P., Mönnich, U. and Morawietz, F. (1999b). Regular description of cross-serial dependencies, *Proceedings of MOL 6*, Orlando, Florida.

Kolb, H.-P., Mönnich, U. and Morawietz, F. (2000). Descriptions of cross-serial dependencies, Ms, University of Tübingen. Under review.

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

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

Mönnich, U. (1997). Adjunction as substitution, *in* G.-J. M. Kruijff, G. Morill and R. Oehrle (eds), *Formal Grammar '97*, Aix-en-Provence, pp. 169–178.

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

Rambow, O. and Satta, G. (1999). Independent parallelism in finite copying parallel rewriting systems, *TCS: Theoretical Computer Science*.

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* C. Retoré (ed.), *Logical Aspects of Computational Linguistics*, Springer, Berlin, pp. 68–95. LNAI 1328.

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

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

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