Workshop TAG+5, Paris, 25-27 May 2000

Derivational Minimalism in Two Regular and Logical Steps

Jens Michaelis and Uwe Mönnich and Frank Morawietz

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

Abstract

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

1. Introduction

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

The work by Michaelis (1999) shows how to encode an arbitrary minimalist grammar (MG) in the sense of Stabler (1997) into a weakly equivalent linear context-free rewriting system (LCFRS). The core idea 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 a corresponding LCFRS represents such an equivalence class, i.e., an infinite set of trees. We take the resulting LCFRSs as our starting point and present in this paper a translation from multiple context-free grammars (MCFGs)—which are a weakly equivalent extension of LCFRSs—into regular tree grammars (RTGs)/monadic second-order (MSO) logic/tree automata. This is done via lifting by viewing MCFG rules as terms in a free Lawvere theory. Since this coding makes projection, tupling and composition explicit, the resulting trees contain these operations as labeled nodes. Therefore we use an MSO transduction—where the regular tree language constitutes the domain—to transform the lifted trees into the intended ones.

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. Thus, the translation from MCFGs to RTGs constitutes a considerable simplification in comparison with other characterizations since it is not built upon derivation trees using productions of the original MCFG as node labels, but rather on the operations of projection, tuple-formation and composition alone.

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

2. Background and Basic Definitions

We first present some basic definitions before we proceed with the actual translation. Let S be a set of sorts. A *many-sorted signature Σ (over S)* 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 $|w|$, where $|w|$ denotes the length of w. Let $X = \{x_1, x_2, x_3, \ldots\}$ be a countable set of variables, and for $k \in \mathbb{N}$ define X_k as $\{x_1, \ldots, x_k\}$. Then, the set of k-ary trees $T(\Sigma, X_k)$ over Σ is built up from X_k using the operators in the usual way: If $\sigma \in \Sigma_{\varepsilon,s} \cup X_k$ for some $s \in S$ and $\varepsilon \in S^*$ with $|\varepsilon| = 0$ then σ is a (trivial) k-ary tree of sort s. If, for some $s \in S$ and $w = s_1 \cdots s_n$ with $s_i \in S$, $\sigma \in \Sigma_{w,s}$ and $t_1, \ldots, t_n$ are k-ary trees with t_i of sort s_i then $\sigma(t_1, \ldots, t_n)$ is a k-ary tree of sort s. Note that $T(\Sigma, X_k) \subseteq T(\Sigma, X_l)$ for $k \leq l$. Let $T(\Sigma, X) = \bigcup_{k \in \mathbb{N}} T(\Sigma, X_k)$.

The 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 $\sigma \in \Sigma_{w,s}$, $\sigma_{\mathbb{A}} : A^w \to A^s$ is a function, where $A^w = A^{s_1} \times \cdots \times A^{s_n}$ and $w = s_1 \cdots s_n$ with $s_i \in S$. The set $T(\Sigma, X)$ can be made into a Σ-algebra $\mathbb{T}$ by specifying the operations as follows. For every $\sigma \in \Sigma_{w,s}$, where $s \in S$ and $w = s_1 \cdots s_n$ with $s_i \in S$, and every $t_1, \ldots, t_n \in T(\Sigma, X)$ with t_i of sort s_i we identify $\sigma_{\mathbb{T}}(t_1, \ldots, t_n)$ with $\sigma(t_1, \ldots, t_n)$.

Our main notion is that of an *algebraic (Lawvere) theory*. 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, where $n \in \mathbb{N}$, $u = u_1 \cdots u_n$ with $u_i \in S$ for $1 \leq i \leq n$ and $v, w \in S^*$,

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

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

For S being a singleton and Σ a (many-sorted) signature over $S^* \times S^*$, the power set $\wp(T(\Sigma, X))$ of $T(\Sigma, X)$ constitutes the central example of interest for formal language theory. The carriers $\langle \wp(T(k, m)) \mid k, m \in \mathbb{N}\rangle$ of the corresponding $S^* \times S^*$-Lawvere algebra are constituted by the power sets of the sets $T(k, m)$, where each $T(k, m)$ is the set of all m-tuples of k-ary trees, i.e. $T(k, m) = \{(t_1, \ldots, t_m) \mid t_i \in T(\Sigma, X_k)\}$.[1] 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 and their connection to linguistics can be found in Mönnich (1998).

[1] Since S is a singleton, S^* can be identified with $\mathbb{N}$, because up to length each $w \in S^*$ is uniquely specified.

A *multiple context-free grammar* (MCFG) is defined as a five-tuple $\mathcal{G} = \langle N, T, F, P, S\rangle$ with N, T, F and P being a finite set of ranked nonterminals, terminals, linear basic morphisms and productions, respectively. $S \in N$ is the start symbol. Each $p \in P$ 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 variables, constants, concatenation, composition and tupling.

A *regular tree grammar* (RTG) is a 4-tuple $\mathcal{G} = \langle \Sigma, \mathsf{F}_0, S, \mathsf{P}\rangle$, where Σ is a many-sorted signature of *inoperatives* and F_0 a set of *operatives* of rank 0. $S \in \mathsf{F}_0$ is the starting symbol and P is a set of productions. Each $p \in \mathsf{P}$ has the form $F \longrightarrow t$, with $F \in \mathsf{F}_0$, and t a term (tree) over $\Sigma \cup \mathsf{F}_0$. An application of a rule $F \longrightarrow t$ "rewrites" F as the tree t. Since RTG rules always just substitute some tree for a leaf-node, it is easy to see that they generate recognizable sets of trees, i.e., context-free string languages (Mezei & Wright 1967).[2]

After these algebraic notions, we briefly present those related to monadic second-order (MSO) logic. MSO logic is the extension of first-order predicate logic with monadic second-order variables and quantification over them. In particular, we are using MSO logic on trees such that individual variables $x, y, \ldots$ stand for nodes in trees and monadic second-order ones $X, Y, \ldots$ for sets of nodes (for more details see, e.g., Rogers 1998).

Before we turn to purely logical notions, we introduce a concept which combines both automata theory and logic. We need a particular type of finite-state automaton: *tree-walking automata with MSO tests* (Bloem & Engelfriet 1997). Intuitively, those automata make transitions from nodes in a tree to other nodes along its branches.

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

The following paragraphs go directly back to Courcelle (1997). Recall that the representation of objects within relational structures makes them available for the use of logical description languages. Let R be a finite set of relation symbols with the corresponding arity for each $r \in R$ given by $\rho(r)$. A relational structure $\mathcal{R} = \langle D_{\mathcal{R}}, (r_{\mathcal{R}})_{r \in \mathcal{R}}\rangle$ consists of the domain $D_{\mathcal{R}}$ and the $\rho(r)$-ary relations $r_{\mathcal{R}} \subseteq D_{\mathcal{R}}^{\rho(r)}$. There does not seem to be a convenient machine model for tree transformations. Fortunately, one can use logic directly to define the desired transduction. The classical technique of interpreting a relational structures within another one forms the basis for MSO transductions. Intuitively, the output tree is interpreted on the input tree. E.g., suppose that we want to transduce the input tree t_1 into the output tree t_2. The nodes of the output tree t_2 will be a subset of the nodes from t_1 specified with a unary MSO relation ranging over the nodes of t_1. The daughter relation will be specified with a binary MSO relation with free variables x

[2]Appropriate definitions for derivations and the tree languages generated can be found in Kolb *et al.* (2000).

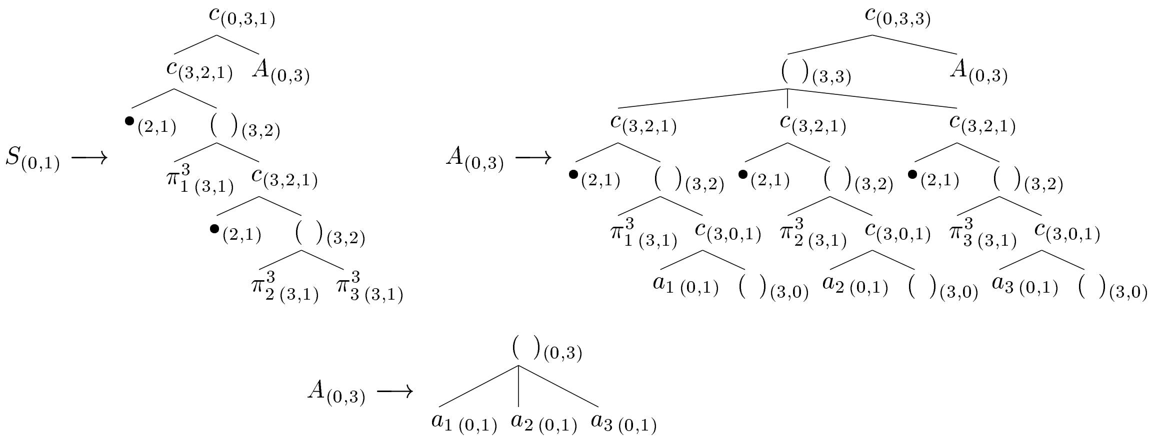

Figure 1: The translated example grammar $\mathcal{G}'$

and y ranging over the nodes from t_1. 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 $(\varphi, \psi, (\theta_q)_{q\in Q})$. It consists of the formulas φ defining the domain of the transduction in $\mathcal{R}$ and ψ defining the resulting domain of $\mathcal{Q}$ and a family of formulas θ_q defining the new relations Q (using only definable formulas from the "old" structure $\mathcal{R}$).

In this sense, our description of non-contextfree phenomena with two devices with only regular power is an instance of the theorem that the image of an MSO-definable class of structures under a definable transduction is not MSO definable in general (Courcelle 1997).

3. Translating MCFGs to RTGs

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

As an example we consider the following MCFG $\mathcal{G} = \langle N, T, F, P, S\rangle$ with $N = \{S, A\}$, $T = \{a_1, a_2, a_3\}$, $F = \{g, h, l\}$ and $P = \{S \to g(A), A \to h(A), A \to l()\}$, where the functions $g : (T^*)^3 \to T^*$, $h : (T^*)^3 \to (T^*)^3$ and $l : (T^*)^0 \to (T^*)^3$ are given by

$$g(x_1, x_2, x_3) = x_1x_2x_3 \qquad h(x_1, x_2, x_3) = (x_1a_1, x_2a_2, x_3a_3) \qquad l(\,) = (a_1, a_2, a_3)$$

The language generated by $\mathcal{G}$ is $\{a_1^n a_2^n a_3^n \mid n > 0\}$.

Now, for $1 \le i \le 3$ let π^3_i denote the i-th projection which maps a 3-tuple of strings from T^* to its i-th component, i.e. a 1-tuple, and let $\bullet$ denote the usual binary operation of concatenation defined for strings from T^*, i.e., $\bullet$ maps a 2-tuple to a 1-tuple. The corresponding (Lawvere) arity of S, a_1, a_2 and a_3 is $(0, 1)$, of A $(0, 3)$, of $\bullet$ (2,1), and the one of π^3_1, π^3_2 and π^3_3 is $(3, 1)$.

Applying the translation T given below to the MCFG $\mathcal{G}$ results in the RTG $\mathcal{G}' = \langle \Sigma, \mathsf{F}_0, S_{(0,1)}, \mathsf{P}\rangle$ with inoperatives $\Sigma = \langle \Sigma_{w,s} \mid w \in (\mathbb{N}\times\mathbb{N})^*, s \in \mathbb{N}\times\mathbb{N}\rangle$, operatives F_0 of rank 0, and productions P which (in tree notation) look as given in Fig. 1. We have $\Sigma_{\varepsilon,(3,0)} = \{(\ )_{(3,0)}\}$, $\Sigma_{\varepsilon,(2,1)} = \{\bullet_{(2,1)}\}$, $\Sigma_{\varepsilon,(0,1)} = \{a_{1\,(0,1)}, a_{2\,(0,1)}, a_{3\,(0,1)}\}$, $\Sigma_{\varepsilon,(3,1)} = \{\pi^3_{1\,(3,1)}, \pi^3_{2\,(3,1)}, \pi^3_{3\,(3,1)}\}$,

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

and $\mathsf{F}_0 = \{S_{(0,1)}, A_{(0,3)}\}$.[3]
As one can see in Fig. 1, the basic functions have been realized as terms with their respective implicit operations as nonterminal (composition and tupling) or terminal (projection and empty tupling) nodes. In the following paragraphs, we sketch the translation T from nonterminal rules of the example MCFG to RTG rules. T takes each rule $X \longrightarrow f(Y)$, where $X, Y \in N$ and $f \in F$, of the MCFG including the corresponding definition of the mapping $f(x_1, \ldots, x_k)$ with $k \geq 0$ and transforms it into a RTG rule as follows. We create a mother node labeled with the appropriate binary composition $c_{(j,k,l)}$ such that the left daughter contains the "lifted" version of $f(x_1, \ldots, x_k)$ under T and the right daughter the translation of the nonterminal Y. Both nonterminals X and Y are used "unchanged", but annotated with the corresponding Lawvere arity resulting in the following schematic presentation of the translation: $X_{(j,l)} \longrightarrow c_{(j,k,l)}(\mathsf{T}(f(x_1, \ldots, x_k)), Y_{(j,k)})$, where f is a mapping from k-tuples to l-tuples of terminal strings.
The easiest case of translating a mapping $f \in F$ from our example via T is the terminal A-rule. We simply view the mapping as a Lawvere term. The function l just returns a triple of a_1, a_2 and a_3. The corresponding tree has a mother node labeled with a ternary tupling symbol and the three unary arguments of the mapping as daughters.[4] The S-rule is more complicated with the function g concatenating three (input) strings. The definition of the function can be written explicitly as the Lawvere term $c_{(3,2,1)}(\bullet, (\ )_{(3,2)}(\pi_1^3, c_{(3,2,1)}(\bullet, (\ )_{(3,2)}(\pi_2^3, \pi_3^3))))$. Note that the implicit binary concatenation $\bullet$ in g now becomes the constant $\bullet_{(2,1)}$. The variables are simply replaced by the projections and concatenated. The resulting term is then applied to the operative $A_{(0,3)}$ such that we get the RHS displayed in the $S_{(0,1)}$-rule in Fig. 1. The recursive case of the A-rule is the most complicated. The mapping returns a triple, so we need a tupling "operator" of appropriate arity $(3, 3)$ as the mother node with 3 daughters. The i-th of its daughters (labeled with $c_{(3,2,1)}$) is built by composing the concatenation constant $\bullet_{(2,1)}$ with the "tupling"-result $(\ )_{(3,2)}$ of the corresponding projection constant $\pi^3_{i(3,1)}$ (which is substituted for the variable x_i) and a particular constant tree. Namely the one which (in terms of the underlying Lawvere algebra) simply "lifts" the constant a_i to the Lawvere-arity of π_i^3 just in order to allow for an appropriate tupling. So, the term (x_1a_1, x_2a_2, x_3a_3) is interpreted as the Lawvere term $(\ )_{(3,3)}(c(\bullet, (\ )_{(3,2)}(\pi_1^3, c(a_1, (\ )_{(3,0)})))), c(\ldots), c(\ldots))$ which appears as the RHS of the corresponding tree grammar rule.
Since RTGs can only generate recognizable (tree) languages, we can characterize them with both MSO logic on trees and tree automata.[5] The tree automaton $\mathfrak{A}_{G'}$ is constructed by transforming the grammar into a normal form such that each RHS is of depth one by introducing auxiliary operatives. Then we can easily construct appropriate transitions by basically reversing the arrow: the nonterminals become state names and the mother node will be read as alphabet symbol. It is know from Thomas (1990) how to transform this tree automaton into an MSO formula $\varphi_{\mathfrak{A}_{G'}}$ by encoding its behaviour. Details for our special case can be found in Kolb *et al.* (2000).

4. Reconstructing 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

[3]For simplicity and readability we will sometimes drop the subscript notion (k, m) from the inoperatives and operatives of rank 0, and sometimes even from the composition symbol $c_{(k,l,m)}$.
[4]Note that we do not need to use a further composition symbol dominating $\mathsf{T}(f)$ in case there is no nonterminal on the RHS of the rule of the MCFG.
[5]An introduction to tree automata can be found in Gécseg & Steinby (1984).

and proper precedence ($\prec$). We will show how to define these relations with an MSO transduction thereby implementing the unique homomorphism mapping the terms into elements of the corresponding regular tree language.. At the core of the transduction is a tree-walking automaton defining the binary relation of immediate dominance ($\triangleleft$) on the nodes belonging to the intended structures. It is based on some simple observations.[6]

1. Our trees feature three families of labels: the "linguistic" symbols L, i.e., the lifted symbols of the underlying MCFG; the "composition" symbols $\mathsf{C} = \{c_{(u,v,w)}\}$; the "tupling" symbols $(\quad)_{(v,u)}$ and the "projection" symbols $\Pi = \{\pi_i^k\}$.
2. All nonterminal nodes in T' are labeled by some $c \in \mathsf{C}$ or a "tupling" symbol. Note that no terminal node is labeled by some c.
3. The terminal nodes in T' are either labeled by some "linguistic" symbol, a "tupling" symbol of the form $(\quad)_{(k,0)}$, i.e. the "empty" tuple, or by some "projection" symbol π_i^k.
4. Any "linguistic" node dominating anything in the intended tree is on some left branch in T', i.e., it is the left daughter of some $c \in \mathsf{C}$ and the sister of a tupling symbol whose daughters evaluate to the intended daughters.
5. For any node ν labeled with some "projection" symbol $\pi_i^u \in \Pi$ in T' there is a unique node μ (labeled with some $c \in \mathsf{C}$) which properly dominates ν and which immediately dominates a node labeled with a "tupling" symbol whose i-th daughter will eventually evaluate to the value of π_i^k. Moreover, μ will be the first node properly dominating ν which is on a left branch and bears a composition symbol. This crucial fact is arrived at by induction on the construction of $\mathcal{G}'$ from $\mathcal{G}$.

By 4. it is not hard to find possible dominees in any T'. It is the problem of determining the actual "filler" of a candidate-dominee which makes up the complexity of the definition of $\triangleleft$. There are three cases to account for:

6. If the node considered carries a "linguistic" label, it evaluates to itself;
7. if it has a "composition" label c, it evaluates to whatever its leftmost daughter evaluates to;
8. if it carries a "projection" label π_i^k, it evaluates to whatever the node it "points to"—by (5.) the i^{th} daughter of a "tupling" node which is dominated by the first C-node on a left branch dominating it—evaluates to.

According to the observations made above, the automaton given in Fig. 2 starts on any node with a "linguistic" label (denoted here by L) which means for the given example $\bullet, a_1, a_2, a_3$. Then it has to go up the first branch, read a composition symbol and descend to its sister. If it reads a "linguistic" node, the automaton stops. If it reads a composition symbol, the automaton goes to the left daughter and tries again. If it reads a tupling symbol, the automaton proceeds with its daughters. On finding a projection symbol, it searches for the appropriate "filler" by going upwards until it is on a leftmost branch which is labeled with a composition symbol. Then it 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.
However, there is another interpretation of such an automaton. Viewed as an ordinary finite-state automaton over the alphabet Δ, $\mathfrak{A}_\triangleleft$ recognizes a regular (string-) language, the *walking language* W which can be translated recursively into an MSO formula $\mathsf{trans}_{W_\triangleleft}$ defining the relation $\triangleleft$ (see Bloem & Engelfriet 1997). We leave the rather tedious process of converting the walking language for the automaton given in Fig. 2 to the reader (a full example of such a conversion can be found in Kolb *et al.* (2000)).

[6]The reader is encouraged to check them against trees T' generated by $\mathcal{G}'$ given in Fig. 1.

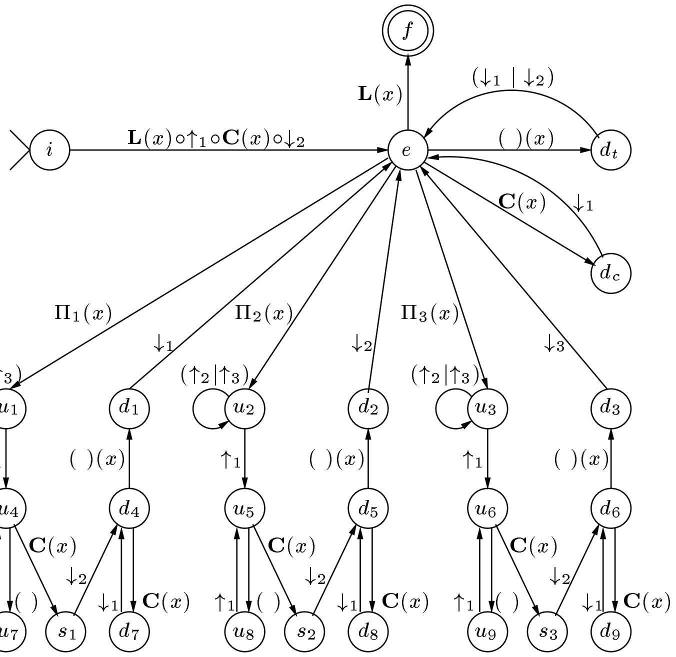

Figure 2: The tree-walking automaton for immediate dominance: $\mathfrak{A}_{\triangleleft}$

To present the actual MSO transduction, we need one further auxiliary definition. It is a well-known fact (e.g. Bloem & Engelfriet 1997) that the reflexive transitive closure R^* of a binary relation R on nodes is (weakly) MSO-definable, if R itself is. This is done via a second-order property which holds of the sets of nodes which are closed under R: $R\text{--}\mathsf{closed}(X) \iff_{def} (\forall x, y)[x \in X \wedge R(x, y) \rightarrow y \in X]$.
Finally, the MSO transduction $(\varphi, \psi, (\theta_q)_{q\in Q})$ with $Q = \{\triangleleft, \triangleleft^*, \triangleleft^+, \prec, \dots\}$ we need to transform the lifted structures into the intended ones is given as follows:

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

As desired, the domain of the transduction is characterized by the MSO formula $\varphi_{\mathfrak{A}_{\mathcal{G}'}}$ 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.

5. Conclusion

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

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

References

BLOEM R. & ENGELFRIET J. (1997). *Characterization of Properties and Relations defined in Monadic Second Order Logic on the Nodes of Trees*. Tech. Rep. 97-03, Leiden University.

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*, p. 313–400. World Scientific.

ENGELFRIET J. (1997). Context-free graph grammars. In G. ROZENBERG & A. SALOMAA, Eds., *Handbook of Formal Languages. Vol. III: Beyond Words*, p. 125–213. Springer.

ENGELFRIET J. & MANETH S. (1999). Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, **154**, 34–91.

GÉCSEG F. & STEINBY M. (1984). *Tree Automata*. Budapest: Akadémiai Kiadó.

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

MEZEI J. & WRIGHT J. (1967). Algebraic automata and contextfree sets. *Information 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. (1998). TAGs M-constructed. In *TAG+ 4th Workshop, Philadelphia*.

RAMBOW O. & SATTA G. (1999). Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, **223**, 87–120.

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

SEKI H., MATSUMURA T., FUJII M. & KASAMI T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, **88**, 191–229.

STABLER E. (1997). Derivational minimalism. In C. RETORÉ, Ed., *Logical Aspects of Computational Linguistics*, p. 68–95, Berlin: Springer. LNAI 1328.

THOMAS W. (1990). Automata on infinite objects. In J. VAN LEEUWEN, Ed., *Handbook of Theoretical Computer Science*, chapter 4, p. 133–191. Elsevier Science Publishers B. V.

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