Well-Nested Tree Languages and Attributed Tree Transducers

Uwe Mönnich
Theoretische Computerlinguistik
Universität Tübingen
Wilhelmstr. 19
72074 Tübingen
uwe.moennich@uni-tuebingen.de

Abstract

Well-nested word languages have been advertised as adequate formalizations of the notion of mild context-sensitivity. The main result of this paper is a characterization of well-nested tree languages in terms of simple attributed tree transducers.

1 Introduction

The intuitive notion of mild context-sensitivity has led to two competing candidates claiming to be an exact formal rendering of the intentions Joshi (1985) was trying to capture when introducing this concept. On the one hand multiple context-free grammars and their equivalents exhibiting a variety of wildly different specifications provide impressive evidence that this precise counterpart of the informal description of mild context-sensitivity constitutes a natural class. On the other hand the well-nested subclass of the multiple context-free grammars has recently been advertised as a formalization more in accordance with Joshi's original intentions ((Kanazawa, 2009), (Kuhlmann, 2007)). Both candidates have counterparts in the realm of tree languages and it is in this context that they are easily recognized as mathematically precise characterizations of two leading linguistic models, minimalist syntax and tree adjoining grammars (cf. (Harkema, 2001), (Michaelis, 2001), (Kepser and Rogers, 2007), (Mönnich, 1997), (Mönnich, 2007)).

The tree languages in question are the multiple regular tree languages ((Raoult, 1997)) and the simple context-free tree languages ((Engelfriet and Maneth, 2000)). Both language families are proper subfamilies of the tree languages generated by context-free hyperedge-replacement graph grammars and the latter family is identical with the output languages of logical tree-to-tree transductions applied to regular tree languages. The obvious question that poses itself is whether the two restricted rule formats or their corresponding tree transducers, finite-copying top-down tree transducers and simple macro tree transducers (MTT), respectively, can be given an equivalent logical characterization in terms of restrictions on the logical formulae defining the relations in the target structures of logical transductions. This is indeed the case. From results presented in (Bloem and Engelfriet, 2000) and in (Engelfriet and Maneth, 1999) it follows that the tree languages which are the output of finite-copying top-down tree transducers applied to regular tree languages are exactly the output tree languages of logical tree transducers which are direction preserving in the sense that edges in the output trees correspond to directed paths in the input trees. As a preliminary step towards an analogous result for languages generated by simple context-free tree grammars (CFT_{sp}) we'll show in this paper that they are equivalent to the output languages of a highly restricted form of attributed tree transducers (ATT). As emphasized by (Bloem and Engelfriet, 2000), attribute grammars are to be regarded as a specification language and, as such, they are much closer to logical specifications than context-free tree grammars.

In previous work Duske et al. (1977) have characterized the inside-out macro languages due to

(Fischer, 1968) as the output string languages of attribute grammars with one synthesized attribute only. In a nutshell, we lift their result to the tree level and restrict it to the case of well-nested tree languages, which are immune to differences of derivation mode ((Kepser and Mönnich, 2006)) and, a fortiori, are inside-out. In their paper mentioned above Engelfriet and Maneth (2000) point out that well-nested tree grammars can be regarded as a notational variant of simple $MTTs$. Our result could accordingly be interpreted as showing the equivalence between simple $ATTs$ and simple $MTTs$. We believe that our construction that avoids the detour via $MTTs$ has the advantage of establishing a direct link between a procedural and a declarative formalization of well-nestedness.

After these introductory remarks we will recall some basic concepts from the theory of tree languages and the theory of attributed tree transducers. The following section is an attempt at reviewing briefly the notion of mild context-sensitivity from the perspective of context-free graph languages. We will then proceed to prove the equivalence of CFT_{sp} and simple $ATTs$ with one attribute only ($1S - ATT_{ss,si}$). We shall close with some remarks regarding the problems to be solved for a logical characterization of CFT_{sp}.

2 Preliminaries

For any set A, A^* is the set of all strings over A. ε is the empty string, $|w|$ is the length of a string w. N denotes the set $\{0, 1, 2, 3, \ldots\}$ of nonnegative integers.

Let S be a finite set of *sorts*. An S-signature is a set Σ given with two mappings $ar : \Sigma \to \mathcal{S}^*$ (the *arity* mapping) and $so : \Sigma \to \mathcal{S}$ (the *sort* mapping). The length of $ar(\sigma)$ is called the *rank* of σ, and is denoted by *rank*(σ). The *type* of σ ($\sigma \in \Sigma$) is the pair $\langle ar(\sigma), so(\sigma)\rangle$. The elements of $\Sigma_{\varepsilon,s}$ are also called constants (of sort s).

In case S is a singleton set $\{s\}$, i.e. in case Σ is a *single-sorted* or *ranked alphabet (over sort s)*, we usually write $\Sigma^{(n)}$ to denote the (unique) set of operators of rank $n \in N$; we also write $\sigma^{(n)}$ to indicate that $rank(\sigma) = n$. In this case, which is of particular interest to us, the set of trees T_Σ is defined recursively as follows. Each constant of Σ, i.e., each symbol of rank 0, is a tree. If σ is of rank k and $t_1, \ldots, t_k$ are trees, then $\sigma(t_1, \ldots, t_k)$ is a tree. A *tree language* $L \subseteq T_\Sigma$ over Σ is a subset of T_Σ. With each tree $t \in T_\Sigma$ we can associate a string $s \in \Sigma_0^*$ by reading the leaves of t from left to right. This string is called the *yield* of t, denoted by $yd(t)$. More formally, $yd(t) = t$ if $t \in \Sigma_0$, and $yd(t) = yd(t_1) \cdots yd(t_k)$ whenever $t = \sigma(t_1, \ldots, t_k)$ with $k \geq 1$. The yield of tree language L is defined straightforwardly as $yd(L) = \{yd(t) | t \in L\}$.

If A is a set (of symbols) disjoint from Σ, then $T_\Sigma(A)$ (alternatively $T(\Sigma, A)$) denotes the set of trees $T_{\Sigma \cup A}$ where all elements of A are taken as constants. Let $X = \{x_1, x_2, x_3, \ldots\}$ be a fixed denumerable set of *input variables* and $Y = \{y_1, y_2, y_3, \ldots\}$ be a fixed denumerable set of *parameters*. Let $X_0 = Y_0 = \emptyset$ and, for $k \geq 1, X_k = \{x_1, \ldots, x_k\} \subset X$, and $Y_k = \{y_1, \ldots, y_k\} \subset Y$. For $k \geq 0, m \geq 0, t \in T_\Sigma(X_k)$, and $t_1, \ldots, t_k \in T_\Sigma(X_m)$, we denote by $t[t_1, \ldots, t_k]$ the result of *substituting* t_i for x_i in t. Note that $t[t_1, \ldots, t_k]$ is in $T_\Sigma(X_m)$. Note also that for $k = 0, t[t_1, \ldots, t_k] = t$.

Definition 1 *A* context-free tree *(CFT) grammar is a tuple $G = (\mathcal{F}, \Omega, S, P)$ where $\mathcal{F}$ and Ω are ranked alphabets of non-terminals and terminals, respectively, $S \in \mathcal{F}^{(0)}$ is the start symbol and P is a finite set of productions of the form*

$$F(y_1, \ldots, y_m) \to \xi$$

where $F \in \mathcal{F}$ and ξ is a tree over $\mathcal{F}$, Ω and Y_m.

If for every $F \in \mathcal{F}^{(m)}$ each $y \in Y_m$ occurs exactly once on the right-hand side of the corresponding rule then the context-free tree grammar is called *simple in the parameters (sp)*. The family of tree languages which is generated by context-free tree grammars which are simple in their parameters is designated as CFT_{sp}.

A grammar $G = \mathcal{F}, \Omega, S, P)$ is called a *regular tree (REGT)* grammar if $\mathcal{F} = \mathcal{F}^{(0)}$, i.e., if all nonterminals are of rank 0.

Attributed tree transducers are a variant of attribute grammars in which all attribute values are trees. Besides *meaning names* which transmit information in a top-down manner, attributed tree transducers contain explicit *context names* which allow information to be passed up from a node to its

mother. Consequently, arbitrary tree walks can be realized by attributed tree transducers.

Definition 2 *An* attributed tree transducer *(ATT) is a tuple*

$$A = (Syn, Inh, \Sigma, \Omega, \alpha_m, R),$$

where Syn and Inh are disjoint alphabets of synthesized *and* inherited *attributes, respectively, Σ and Ω are ranked alphabets of* input *and* output *symbols, respectively, α_m is a synthesized attribute, and R is a finite set of rules of the following form: For every $\sigma \in \Sigma^{(m)}$, for every $(\gamma, \rho) \in ins_\sigma$ (the set of inside attributes of σ), there is exactly one rule in R_σ:*

$$(\gamma, \rho) \to \xi$$

where $\xi \in T_{\Omega \cup out_\sigma}$ and out_σ is the set of outside attributes of σ.

Definition 3 *For every $\sigma \in \Sigma^{(m)}$, the set of* inside *attributes is the set $ins_\sigma = \{(\alpha, \pi) | \alpha \in Syn\} \cup \{(\beta, \pi i) | \beta \in Inh, i \leq m\}$ and the set of* outside *attributes is the set $out_\sigma = \{(\beta, \pi) | \beta \in Inh\} \cup \{(\alpha, \pi i) | \alpha \in Syn, i \leq m\}$. π and ρ are path variables ranging over node occurrences in the input tree.*

$ATTs$ with rules R_σ at an input symbol σ in which each outside attribute occurs exactly once are called *simple* attributed tree transducers. We denote this class by $ATT_{ss,si}$ on the model of macro tree transducers that are simple, i.e., linear and nondeleting, both in their input variables and in their parameters ($MTT_{si,sp}$).

The *dependencies* between attribute occurrences in an input tree s can be represented with the help of R_σ. An instance of an attribute occurrence (α, π) depends on another occurrence (α', π') if σ labels node u in s, R_σ contains the rule $(\alpha', \pi') \to \xi$ and (α, π) labels one of the leaves in ξ.

The *dependency graph* $D(s)$ of an input tree $s \in T_\Sigma$ consists of the set of attribute occurrences together with the dependencies according to the rules in R. Reversing the direction of these dependencies leads to the notion of a *semantic graph* $S(s)$ of an input tree $s \in T_\Sigma$.

An attributed tree transducer is *noncircular* if the paths of attribute dependencies are noncircular. It is well known that noncircular ATTs have unique *decorations dec*, functions which assign each attribute occurrence a tree over $\Omega \cup out_\sigma$ in accordance with the productions R_σ.

Definition 4 *The* transduction *realized by a noncircular attributed tree transducer A is the function*

$$\tau_A = \{(s, t) | s \in T_\sigma, t \in T_\Omega, t = dec_s(\alpha_m, \epsilon)\}$$

Remark 1 *ATTs which have synthesized attributes only are very close to top-down tree tranducers. Their rules*

$$(\gamma, \rho) \to \xi \text{ in } R_\sigma$$

correspond to tree transducer rules

$$\alpha(\sigma(x_1, \ldots, x_m)) \to \xi'$$

where $\sigma \in \Sigma^{(m)}$ and ξ' is obtained from ξ by substituting every $(\alpha, \pi i)$ by $\alpha(x_i)$.

3 Mild Context-Sensitivity

Even though the list of characteristic properties Joshi (1985) has suggested as characteristic features of a linguistically adequate extension of context-freeness can hardly be considered as a formally explicit definition it has become accepted as an informal description of a precise counterpart that has turned out to be remarkably stable with respect to a variety of wildly different specifications. This stability provides impressive evidence that the precise counterpart of the informal description of mild context-sensitivity constitutes a natural class. Over the last couple of years the formal characterizations assembled in Tabel 1, which all satisfy the criteria for membership in the class of mildly context-sensitive languages, have turned out to be weakly equivalent.

Besides this group of attempts at providing a formal explication for a framework adapted to the level of complexity found in natural language syntax another proposal for an exact definition of mild context-sensitivity is currently the focus of attention among linguists. We referred above to Kanazawa's claim that the so-called well-nested multiple context-free word grammars constitute a more faithful approximation of Joshi's original ideas than the multiple context-free grammars that were

$MCFL$	=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
$STR(HR)$	=languages generated by string generating hyperedge replacement grammars
$OUT(DTWT)$	=output languages of deterministic tree-walking tree-to-string transducers
$ydDT_{fc}(REGT)$	=yields of images of regular tree languages under deterministic finite-copying top-down tree transduction
$PGRPT$	=languages defined by pregroups with tupling

Table 1: Classes of mildly context-sensitive word languages

the most prominent member of the first mathematically precise rendering of the notion of mild context-sensitivity. Well-nested multiple context-free string grammars satisfy the additional condition of a matching-like or well-nested rearrangement of the arguments of non-terminals on the right-hand sides of possible rules allowed by this type of grammar. Furthermore, it is obvious that well-nested multiple context-free languages are included in the multiple context-free languages in general. Besides this formal characterization in terms of well-nested multiple context-free languages, a number of further precise representatives of this second version of mild context-sensitivity have been found. This group assembled in Table 2 may not be as impressive as the long list of equivalent formalizations that was listed in connection with the first proposal for a formal rendering of mild context-sensitivity. Still, the equivalence of such wildly different systems as abstract categorial grammars and (subclasses of) macro tree transducers cannot be a pure coincidence, but has to be accepted as a sign of an underlying conceptual coherence.

The Table 2 contains besides examples of well-nestedness on the level of word languages two specimens of well-nested tree languages. Needless to say, the derived tree structures are well-nested in both alternatives suggested as formal renderings of mild-context-sensitivity. The well-nestedness condition concerns exclusively the rule format. In sharp contrast with the situation on the word level the multiple regular tree languages and the simple context-free languages mentioned in the introduction as candidate formalizations of minimalist syntax and tree adjoining grammars, respectively, are proper subfamilies of the tree languages gen-

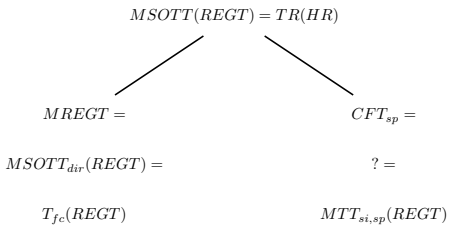

Figure 1: Classes of mildly context-sensitive tree languages

erated by context-free hyperedge-replacement graph grammars $(TR(HR))$ and the latter family is identical with the output languages of logical tree-to-tree transductions applied to regular tree languages $(MSOTT(REGT))$. Both mildly context-sensitive subfamilies of tree languages can be given descriptions in terms of grammatical systems and syntax-directed tree transducers. To repeat their formal characterizations, the multiple regular tree languages are exactly the output tree languages of finite-copying top-down tree transducers (T_{fc}) which, in turn, are the same as the output tree languages of direction preserving logical transductions $(MSOTT_{dir})$. The simple context-free tree languages, analogously, are exactly the output tree languages of simple macro tree transducers which, in turn, are the same as the output tree languages of simple attributed tree transducers with one synthesized attribute only, as shown in this paper. Their logical characterization is still open. Figure 1 summarizes the overview of the previous discussion concerning the inclusion relation among mildly context-sensitive tree languages.

Given the incomparability of the two families of mildly context-sensitive tree languages, the investi-

$MCFL_{wn}$	= well-nested multiple context-free languages
MAC_{sp}	= simple macro languages
$ydCFT_{sp}$	= yields of context-free tree languages
$ydMTT_{si,sp}(REGT)$	= yields of output languages of tree transductions realized by simple macro tree transducers taking regular tree languages as input
CFT_{sp}	= simple context-free tree languages
$ACG_{(2,3)}$	= abstract categorial languages with abstract vocabulary of order at most 2 and lexical images of atomic types of order at most 3

Table 2: Classes of well-nested mildly context-sensitive word and tree languages

gation of their formal properties is of obvious interest for the evaluation of the corresponding linguistic frameworks. Should it turn out that ill-nested structures are needed for an adequate representation of structures exemplified in natural syntax the theory of "mildly context-sensitive" graph languages might still be the framework of choice as long as these structures are more tree- than graph-like.

Trees can be uniquely characterised by specifying their hierarchical structure, they can be enumerated by means of a production system and they can be fed into an abstract automaton. For graphs no suitable notion of finite automaton has been proposed. Similarly, Graph grammars, on the other hand, do not generate their output in a way that is patterned by an independently given hierarchical structure which would be intrinsic to this data type.

In spite of this difference of the two data structures, the fact that certain linguistic phenomena are graphs does not as such preclude the possibility of an approach which sees them as the result of a logically defined tree transduction. Bauderon and Courcelle (1987) present an approach to graphs by defining a (generally infinite) set of graph operations such that every finite graph can be constructed by finitely many applications of these graph operations. For the particular licensing theory, head-driven phrase structure grammar ($HPSG$), we have shown ((Kepser and Mönnich, 2003)) that the classes of finite graphs defined by a finite $HPSG$ signature and grammar are indeed such that they cannot be generated by a finite set of graph operations. But this purely metatheoretic limitation of $HPSG$ in general is not supported by the amount of tree-likeness retained in concrete analysis structures proposed for grammatical phenomena in languages like Serbo-Croatian. It is thus not excluded that the general approach in terms of syntax-directed morphisms by means of tree-to-tree or tree-to-(tree-like) graph transducers can also be applied to a conceptual framework in which ill-nested derived structures enjoy full citizenship.

4 Main Result

In this section we will present our main result. The proof requires two constructions that reduce simple context-free tree grammars (CFG_{sp}) to simple attributed tree transducers with one synthesized attribute only ($1S - ATT_{ss,si}$) and vice versa.

4.1 From CFT_{sp} to $1S - ATT_{ss,si}$

Before showing that for every CFG_{sp} G an $1S - ATT_{ss,si}$ A_G can be constructed with the same output language we first regularize G by translating its rules into terms over a new signature that is obtained from the original one by a so-called lifting process (cf. (Maibaum, 1974)). It would have been possible to reduce a given CFG_{sp} directly into an equivalent $1S - ATT_{ss,si}$. We have opted for the indirect approach, because the lifting technique is closely related to a decomposition result for $MTTs$ to the effect that the transformation performed by this family of tree transducers can be split into a symbolic replacement and a second-order substitution (cf. (Engelfriet and Vogler, 1985)). We shall see in the last section that the recourse to context attributes in coping with the second-order substitution cannot be avoided, destroying the prospect of a characterization of well-nestedness by means of straightforward tree homomorphisms.

Definition 5 (LIFT) *Let Σ be a ranked alphabet and X a finite set of variables. The* derived *N-sorted alphabet Σ' is defined as follows: For each $n \geq 0$,*

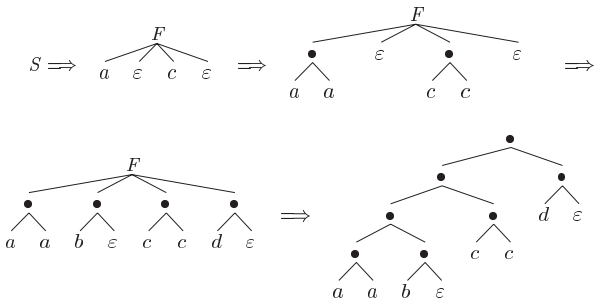

Figure 2: Derivation of $aabccd$.

$\Sigma'_{\varepsilon,n} = \{f' \mid f \in \Sigma_n\}$ *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* i*th* 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* $k_1 = \ldots = k_n = k$.

For $k \geq 0$, $LIFT^{\Sigma}_k : T(\Sigma, X_k) \to T(\Sigma', k)$ *is defined as follows:*

$$LIFT^{\Sigma}_k(x_i) = \pi^k_i$$

$$LIFT^{\Sigma}_k(f) = c_{0,k}(f') \; for \; f \in \Sigma_0$$

$\text{LIFT}^{\Sigma}_k(f(t_1, \ldots, t_n)) =$
$c_{n,k}(f', LIFT^{\Sigma}_k(t_1), \ldots, LIFT^{\Sigma}_k(t_n)) \; for \; f \in \Sigma_n$, $n \geq 1$

By way of example, we consider the effect of the lifting transformation on a CFT_{sp} G that generates the language $L(G) = \{a^n b^m c^n d^m\}$, where we have omitted the concatenation symbols. The grammar consists of the terminal alphabet $\Omega = \{\varepsilon^{(0)}, a^{(0)}, b^{(0)}, c^{(0)}, d^{(0)}, \bullet^{(2)}\}$, the non-terminal alphabet $\mathcal{F} = \{S^{(0)}, F^{(4)}\}$and the following group P of rules:

$$\begin{aligned}
S &\to \varepsilon \\
S &\to F(a, \varepsilon, c, \varepsilon) \\
S &\to F(\varepsilon, b, \varepsilon, d) \\
F(x_1, x_2, x_3, x_4) &\to F(\bullet(a, x_1), x_2, \bullet(c, x_3), x_4)) \\
F(x_1, x_2, x_3, x_4) &\to F(x_1, \bullet(b, x_2), x_3, \bullet(d, x_4)) \\
F(x_1, x_2, x_3, x_4) &\to \bullet(\bullet(\bullet(x_1, x_2), x_3), x_4)
\end{aligned}$$

Figure 2 illustrates a derivation of the string $aabccd$ with explicit concatenation.

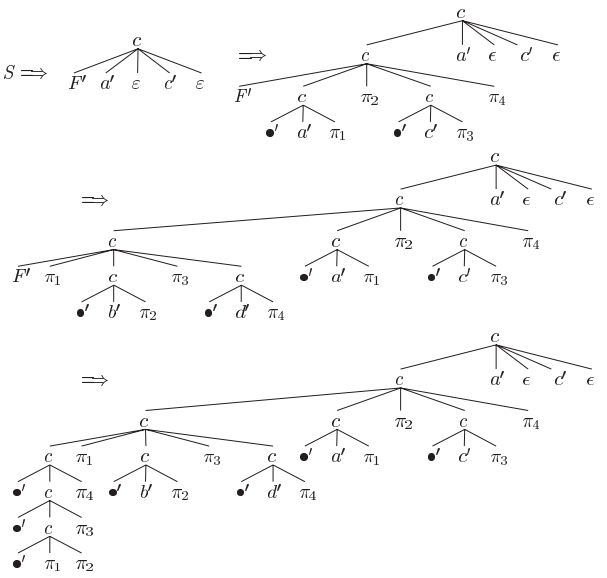

Figure 3: Dervation of lifted $aabccd$.

The process of applying the lifting transformation to the grammar G given above returns a regular tree grammar G' with the following group of rules P':

$$\begin{aligned}
S' &\to \varepsilon \\
S' &\to c(F', a', \varepsilon, c', \varepsilon) \\
S' &\to c(F', \varepsilon, b', \varepsilon, d') \\
F' &\to c(F', c(\bullet', a', \pi^4_1), \pi^4_2, c(\bullet', c', \pi^4_3), \pi^4_4) \\
F' &\to c(F', \pi^4_1, c(\bullet', b', \pi^4_2), \pi^4_3, c(\bullet', d', \pi^4_4)) \\
F' &\to c(\bullet', c(\bullet', c(\bullet', \pi^4_1, \pi^4_2), \pi^4_3), \pi^4_4)
\end{aligned}$$

We parallel the derivation of the example string $aabccd$ with the corresponding derivation in the lifted grammar G' in Figure 3.

The terminal tree in Fig 3 over the lifted signature is yet to be translated into the structures generated by the CFT_{sp} G. These latter structures are coded into the lifted trees and are easily retrieved by unraveling the information contained in the projection and composition symbols. Algebraically speaking, the step from the lifted to the intended structures is nothing but the value of the unique homomorphism from the free term algebra into the substitution algebra over the lifted signature. This homomorphism is simulated by the attributed tree transducer which constitutes the core of the proof of the crucial lemma.

Lemma 1 *For every* CFT_{sp} G*, there is an* $1S - ATT_{ss,si}$ A_G *that outputs the same language when applied to the lifted derivation trees of* G'.

Proof For a given CFG_{sp} $G = (\mathcal{F}, \Omega, S, P)$ an $1S - ATT_{ss,si} = (Syn, Inh, \Sigma, \Omega, \alpha_m, R)$ A_G that outputs the same language is defined from the components of Gs lifted version in the following way.

- $Syn = \{\alpha\}$ with $\alpha = \alpha_m$ at the root node
- $Inh = \{\beta_j | 1 \leq j \leq m\}$ with m the maximal sort of a lifted non-terminal F'
- Every symbol in the derivation trees is assigned one synthesized attribute.
- Every projection symbol is assigned one inherited attribute
- Every lifted terminal symbol f' of sort n is assigned n inherited attributes.
- The function ass_{inh} assigning inherited symbols to composition symbols c is defined by structural induction on the trees on the rhs of P':

  $$ass_{inh}(t) = proj(ass_{inh}(t_1)$$

  where $t = c(t_1, \ldots, t_n)$ and where $proj$ deletes all inherited attributes of t_1 that have no counterparts in $ass_{inh}(t_i)$ $(2 \leq i)$.
- Each $R_{\pi_i^n}$ contains the rule

  $$\alpha\pi \rightarrow \beta_i\pi$$

- Each $R_{f'}$ contains the rule

  $$\alpha\pi \rightarrow f(\beta_1, \ldots, \beta_n)$$

  where n is the sort of f'.
- Each R_c contains the rules

  $$\begin{array}{rcl} \alpha\pi & \rightarrow & \alpha\pi 1 \\ \beta_i\pi 1 & \rightarrow & \alpha\pi i + 1 \\ \beta_i\pi j & \rightarrow & \beta_i\pi \end{array}$$

  provided that the inherited attributes belong to the outside attributes of the composition symbol c.

The correctness of the construction follows from an easy rule induction.

Remark 2 *In the proof of Lemma 1 we have slightly departed from the official definition of an ATT by assigning inherited attributes to composition symbols in a context-dependent way. This could be repaired by labeling composition symbols with the non-terminals on the lhs of P.*

By way of example, consider the slightly simplified lifted tree in Figure 4 which is decorated with both the synthesized and inherited attributes and their values. Evaluating the meaning attribute α_m by traveling along the semantic graph results in the output tree $\bullet(\bullet(\bullet(\bullet(a, a), \varepsilon), \bullet(c, c)), \varepsilon)$.

4.2 From $1S - ATT_{ss,si}$ to CFG_{sp}

It was mentioned above that attributed tree transducers are attribute grammars with all their attribute values restricted to trees and their semantic functions to substitution of trees for dependent leaves. Second–order substitution for internal nodes of trees is achieved through the upward information transport that is made possible by the inherited attributes. Integrating this information transfer with the leaf substitution leads to the kind of substitution or adjunction, for that matter, characteristic of context-free grammars.

Lemma 2 *For every $1S - ATT_{ss,si}$ A, there exists an equivalent CFG_{sp} G_A.*

Proof Let $A = (Syn, Inh, \Sigma, \Omega, \alpha_m, R)$ be an attributed tree transducer with one synthesized attribute only such that each outside attribute at an input symbol σ occurs exactly once in R_σ. An equivalent simple context-free tree grammar G_A is defined as follows:

- $\mathcal{F} = \Sigma$ where the arity of non-terminals is given by the number of interited attributes assigned to them in the input tree.
- $\Omega = \Omega$
- $S = \{\sigma^{(0)}\}$ with $\sigma \in \Sigma$ labeling the root of an input tree.
- For every $\sigma \in \Sigma$ we construct a rule

  $$\sigma(x_1, \ldots, x_n) \rightarrow t$$

  where $t = COMP(\xi)$ and ξ is the right-hand side of the only synthesized attribute α in R_σ.

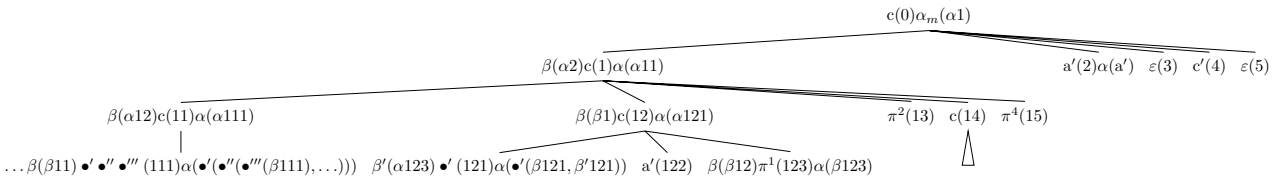

Figure 4: Evaluation of lifted sample tree

The right-hand sides of rules in R_σ are designated by $rhs(\gamma\pi, \sigma)$ in the following:

(i) If $\xi = \alpha\pi i$ then

$$COMP(\xi) = \rho(t_1, \ldots, t_m)$$

where ρ labels the ith daughter of σ, m is the arity of ρ and

$$t_j = COMP(rhs(\beta_j\pi i, \sigma))$$

(ii) If $\xi = \beta_j\pi$ then

$$COMP(\xi) = x_i$$

(iii) If $\xi = f(\xi_1, \ldots, \xi_r)$ for $f \in \Omega^{(r)}$

$$COMP(\xi) = f(COMP(\xi_1), \ldots, COMP(\xi_r))$$

By a routine inspection it is easy to verify that the resulting grammar G_A is indeed simple and that it generates exactly the output language of A.

Remark 3 *The construction in the preceding lemma is inspired by the proofs of Lemma 6.1 in (Engelfriet and Maneth, 1999) and of Lemma 5.11 in (Fülöp and Vogler, 1998). In these references it was shown how to turn an ATT into an equivalent MTT. MTTs can be regarded as tree grammars in which the generation process is controlled by the input trees. In the present case these input structures assume the role of derivation trees. This leads to a certain amount of proliferation in the rule component of the resulting grammar. It is not difficult to prove that this rule proliferation is of no consequence for the derived tree language, i.e., the derived structures.*

By Lemmas 1 and 2 we obtain our main result.

Theorem 1 *The well-nested tree languages are exactly the output languages of simple attributed tree transducers with one synthesized attribute only applied to regular tree languages.*

5 Envoi

Bloem and Engelfriet (2000) have shown that the output language of single use attributed tree transducers applied to regular tree languages are equal to the translations definable by monadic second-order logic. By the result presented in this paper this could be extended to a logical description of simple context-free tree languages. The question, though, of how to give a logical characterization for this family of tree languages is still open. The same question regarding the special case of monadic context-free tree languages was answered in (Mönnich, 2008). The main theorem of that talk states that the edge definitions of a logical characterization with respect to these languages are either direction preserving or direction reversing. This characterization depends crucially on the possibility of proving a Greibach normal form for monadic simple context-free grammars, a result that would require a completely different proof in the case of non-monadic alphabets.

An alternative way out by means of a bimorphic characterization is not open, either. Bimorphisms were introduced by Nivat (1968) for word languages and later extended to trees by Arnold and Dauchet (1982). A bimorphism is a triple $B = (\phi, L, \psi)$ where L is a regular language and ϕ and ψ are homomorphisms. A bimorphism induces a relation B

defined as $B = \{(\phi(t), \psi(t) \mid t \in L\}$. It is interesting to note that the relations specified by tree transducers are captured by bimorphisms in which ψ is unrestricted and ϕ is a relabeling. Shieber (2004) has shown that the relations defined by synchronous tree-substitution grammars correspond exactly to bimorphism relations in which both homomorphisms are linear and non-deleting. An analogous result is not possible for simple context-free tree grammars. Take the familiar example of a simple context-free tree grammar G over a monadic alphabet that generates the language $L = \{a^n(b^n(e))\}$. According to a well-known theorem due to Rounds (1970) every monadic output tree language of a tree transducer applied to a regular language is regular. Consequently, there is no hope of giving a characterization of context-free tree languages without having recourse to context attributes enabling an upward information transport in the controlling input tree.

References

André Arnold and Max Dauchet. 1982. Morphismes et bimorphismes d'arbres. *Theoretical Computer Science*, 20:33–93.

Michel Bauderon and Bruno Courcelle. 1987. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20:83–127.

Roderick Bloem and Joost Engelfriet. 2000. A Comparison of Tree Transductions Defined by Monadic Second-Order Logic and by Attribute Grammars. *J. Comp. System Sci.*, 61:1–50.

J. Duske, R. Parchmann, M. Sedello, and J. Specht. 1977. Io-macrolanguages and attributed translations. *Information and Control*, 35:87–105.

Joost Engelfriet and Sebastian Maneth. 1999. Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations. *Information and Computation*, 154:34–91.

Joost Engelfriet and Sebastian Maneth. 2000. Tree Languages Generated by Context-Free Graph Grammars. In Hartmut Ehrig et al., editor, *Graph Transformation*, number 1764 in LNCS, pages 15–29. Springer.

Joost Engelfriet and Heiko Vogler. 1985. Macro tree transducers. *Journal of Computer and System Sciences*, 31(1):71–146.

Michael J. Fischer. 1968. Grammars with Macro-Like Productions. In *Proc. 9th IEEE Symp. on Switching and Automata Theory*, pages 131–142.

Zoltán Fülöp and Heiko Vogler. 1998. *Syntax-Directed Semantics - Formal Models Based on Tree Transducers*. Springer, New York and Berlin.

Hendrik Harkema. 2001. *Parsing Minimalist Languages*. Ph.D. thesis, University of California at Los Angeles.

Aravind Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions. In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press.

M. Kanazawa. 2009. The convergence of well-nested mildly context-sensitive grammar formalisms. In *14th Conference on Formal Grammar*. Slides available at http://research.nii.jp/~kanazawa/.

Stephan Kepser and Uwe Mönnich. 2003. (Un-)Decidability results for head-driven phrase structure grammar. In Giuseppe Scollo and Anton Nijholt, editors, *Proceedings Algebraic Methods in Language Processing (AMiLP-3)*, pages 141–152.

Stephan Kepser and Uwe Mönnich. 2006. Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science*, 354:82–97.

S. Kepser and J. Rogers. 2007. The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. In M. Kracht, G. Penn, and E. Stabler, editors, *Mathematics of Language 10*.

M. Kuhlmann. 2007. *Dependency Structures and and Lexicalized Grammars.* Ph.D. thesis, Saarland University.

Thomas S. E. Maibaum. 1974. A generalized approach to formal languages. *Journal of Computer and System Sciences*, 8(3):409–439.

Jens Michaelis. 2001. *On Formal Properties of Minimalist Grammar*, volume 13 of *Linguistics in Potsdam*. Universität Potsdam.

Uwe Mönnich. 1997. Adjunction as substitution. In G-J. M. Kruijff, G. Morill, and R. Oehrle, editors, *Formal Grammar '97*, pages 169–178.

U. Mönnich. 2007. Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions. In J. Rogers and S. Kepser, editors, *Proceedings Model Theoretic Syntax at 10*.

Uwe Mönnich. 2008. How to Church a Joshi. In *TAG+9*.

Maurice Nivat. 1968. Transductions des langages de chomyky. Thèse d'état, Paris 1968.

Jean-Claude Raoult. 1997. Rational Tree Relations. *Bull. Belg. Math. Soc.*, 4:149–176.

William C. Rounds. 1970. Mappings and grammars on trees. *Mathematical Systems Theory*, 4(3):257–287.

Stuart M. Shieber. 2004. Synchronous grammars as tree transducers. In *Proceedings TAG+7*.