

Logical Definitions of Mildly Context-Sensitive Grammar Formalisms

Uwe Mönnich

Theoretische Computerlinguistik
Universität Tübingen
`uwe.moennich@uni-tuebingen.de`

Dagstuhl, May 8, 2013



- Introduction: Mild Context-Sensitivity and MSO Transductions
- Minimalist Grammars as Direction Preserving MSO Tree Transductions
- Logical Characterization of Extended Tree Adjoining Grammars
- Set-Local Multicomponent Tree Adjoining Grammars as Minimalist Tree Adjoining Grammars
- Envoi



Joshi(1985)

- Inclusion of context-free languages
- Parsing problem solvable in polynomial time
- Constant growth
- Upper bound on cross-serial dependencies



Discussion

Constant Growth (Weir 1988) A tree or word language L has the property of *constant growth* if there is a constant c_0 and a finite set of constants C such that for all $w \in L$ where $|w| > c_0$ there is a $w' \in L$ such that $|w| = |w'| + c$ for some $c \in C$.

Upper Bound on Cross-Serial Dependencies A tree or word language accounts for a bounded amount of *cross-serial dependencies* if there is a constant c such that at most c dependencies can be coordinated between different branches of a derivation tree. (Drewes 1999)

A tree or word language L accounts for a bounded amount of *cross-serial dependencies* if there is a constant $c \geq 2$ such that $\{w^k | w \in T^*\} \in L$ for all $k \leq c$. (Kallmeyer 2010)

Discussion

- 1 Mildly context-sensitive grammars are slightly more powerful than context-free grammars. They appear, nevertheless, adequate for characterizing most phenomena in natural languages that cannot be captured by context-free grammars.
- 2 Natural languages exhibit at least some constructions which lead to (i) non-CF string languages, or (ii) to non-CF, i.e., non-recognizable structures, even though the resulting string languages are formally context-free. Both phenomena show up in the West-Germanic languages: the verbal complex of Züritütsch is an example of (i), while (ii) is exhibited—for different reasons—by the corresponding constructions of Dutch and Standard German.
- 3 Minimalist, Tree Adjoining and Multicomponent Tree Adjoining grammars are mildly context-sensitive.



Example

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

German: Palindrome language—CF

(... omdat) Karel Marie Piet Jan liet helpen leren zwemmen

Dutch: $a^n b^n$ —CF

(... wil) mer de maa em chind lönd hälffe schwüme



Swiss-German: $a^n b^m c^n d^m$ —Non-CF



Fundamental Properties of Semantic Interpretation

- The method of semantic interpretation defines relational target structures **inside** (a bounded number of copies of) input relational structures.
- The size increase of the output structure is therefore **linear** in the size of the input structure.
- This accounts for the linguistically significant fact that the syntax models falling within the spectrum of MSO transducers are **mildly context-sensitive**.



Useful Properties of Semantic Interpretation

- (i) Grammar independence
- (ii) Closure under Boolean operations
- (iii) Decidability
- (iv) Modularity
- (v) Control over consistency
- (vi) Connection with descriptive complexity theory
- (vii) Mild Context-Sensitivity
- (viii) Structure independence
- (ix) Faithfulness to licensing theories
- (x) Queries definable in MTS



Definition

Given two ranked alphabets Σ and Ω and a finite set C of copy names, a *monadic second-order definable tree to tree transducer (MSOTT)* T from T_Σ to T_Ω is specified by the following formulas of the monadic second-order language over Σ :

- (i) a closed formula φ , the *domain* formula
- (ii) formulas $\nu_c(x)$ with $c \in C$, the *node* formulas
- (iii) formulas $\psi_{\delta,c}(x)$ with $c \in C$ and $\delta \in \Omega$, the *labelling* formulas
- (iv) formulas $\chi_{i,c,d}(x,y)$ with $c, d \in C$ and $i \leq$ maximal arity of symbols in Ω , the *edge* formulas



MSO Tree Transducers: Special Cases

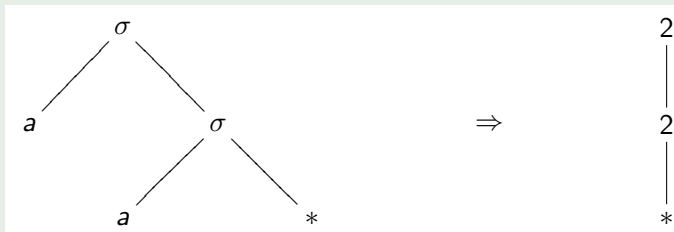
In logical transducers **arbitrary paths** in input trees can serve as definitions of the edges of output trees. Of particular interest regarding the logical analysis of (formalizations of) theories of **natural language syntax** is the fact that natural restrictions on path definitions correspond to current models of syntax. Multiple regular tree grammars, a faithful rendering of **minimalist syntax**, correspond to **direction preserving** MSO tree to tree transducers (*MSOTTs*). Monadic simple context-free tree grammars, a faithful rendering of extended *TAGs*, correspond to *MSOTTs* that **preserve** or **reverse** the direction of the paths in input trees which define the edges of output trees. This family of tree transducers is designated by $MSOTT_{dir,rev}$. We depart slightly from this definition in allowing defining upwards paths between a leaf node and the daughter of a dominating branching node. In this situation we speak of a **slight modification** of $MSOTT_{dir,rev}$.



Example (Fülöp, Vávölgyi)

Example

This is a direction preserving transducer that transforms a tree over $\Sigma = \{\sigma^{(2)}, a^{(0)}, *^{(0)}\}$ into a tree over $\Omega = \{1^{(1)}, 2^{(1)}, a^{(0)}, *^{(0)}\}$ which encodes the path leading from the root of the input tree to the leaf labeled $*$ if there is exactly one such leaf, and results in the single tree node a , otherwise.



Example

Node and Edge Formulas

$$\psi_1(x) = us \wedge \exists y, z(\text{edg}_1(x, y) \wedge \text{path}(y, z) \wedge \text{lab}_*(z)),$$

$$\psi_2(x) = us \wedge \exists y, z(\text{edg}_2(x, y) \wedge \text{path}(y, z) \wedge \text{lab}_*(z)),$$

$$\psi_*(x) = us \wedge \text{lab}_*(x),$$

$$\psi_a(x) = \neg us \wedge \text{root}(x),$$

$$\chi_1(x, y) = \text{edg}(x, y),$$

where 'us' stands for $\exists z(\text{lab}_*(z) \wedge \forall y(\text{lab}_*(y) \rightarrow y = z))$.



Theorem (Bloem, Courcelle, Drewes, Engelfriet, Maneth)

$$\begin{aligned} TR(HR) &= TR(NR) = MSOTT(REGT) = ATT_{sur}^R(REGT) \\ &= MTT_{fc}(REGT) = T_{fc} \circ MTT_{si,sp}(REGT) \end{aligned}$$



The family of tree languages generated by **context-free graph grammars** $TR(HR)$ and $TR(NR)$ can be characterized in terms of a two-step process: the output of **finite-copying top-down tree transducers** T_{fc} is evaluated by means of a second-order tree substitution which is carried out in an algebra of graph operations. In addition, this particular tree substitution can be performed by a **macro tree transducer** $MTT_{si,sp}$ that is both simple in the input and simple in the parameters. Coupled with the insight that applied to the class of **regular tree languages** $REGT$ the class of **finite-copying macro tree transducers** MTT_{fc} and the class of **single use restricted attributed tree transducers** with relabeling ATT_{sur}^R result in the same class of output languages, the class of output languages of **tree to tree transduction definable in monadic second-order logic** $MSOTT(REGT)$, this leads to the string of equivalences of the theorem.



The equivalence of the theorem

$$MSOTT(REGT) = T_{fc} \circ MTT_{si,sp}(REGT)$$

transfers the classical decomposition result for macro tree transducers into the well-behaved realm of mildly context-sensitive tree languages. The constituent elements of this decomposition correspond to a large extent to formalizations of two of the leading contemporary theories of natural language syntax, viz. [Minimalist Syntax](#) and [Tree Adjoining Grammar](#). Their composition reflects the intuition behind the [Multicomponent Tree Adjoining Grammar](#).



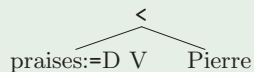
Minimalist Grammars

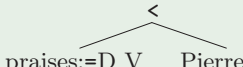
- Generative theory
- Lexically driven
- Feature consuming

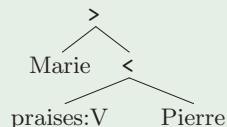


Example

praises::=D =D V + Pierre::D ⇒



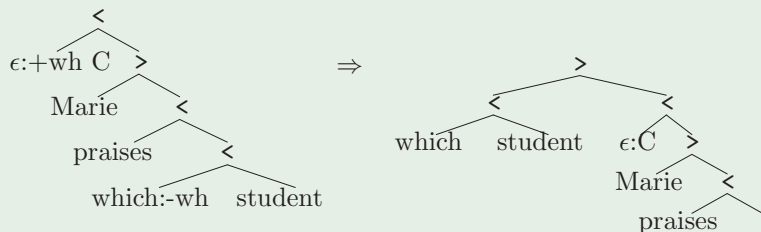
 + Marie::D ⇒



Merge (Stabler 2011)



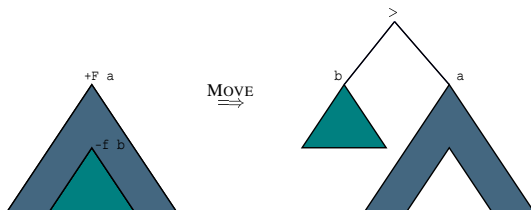
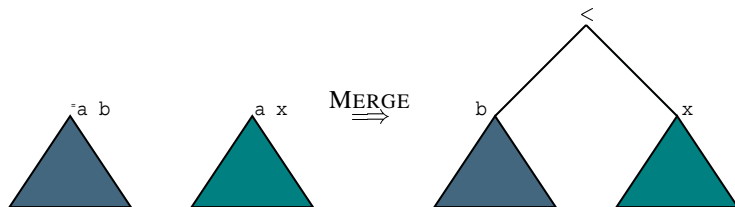
Example



Move (Stabler 2011)



Merge / Move



Minimalist Grammars (Stabler 1997)

$$\mathcal{MG} = \langle \text{NonSyn}, \text{Syn}, \text{Lex}, F \rangle$$

<i>NonSyn</i>	=	<i>Phon</i> \cup <i>Sem</i>
<i>Syn</i>	=	base \cup select \cup licensee \cup licensor
<i>Lex</i>	=	a finite set of trees
<i>F</i>	=	{merge, move}
<i>Phon</i>		are the phonological features
<i>Sem</i>		are the interpreted/semantical features
base	=	{c, t, v, d, n, ...}
select	=	{=x, =X, X= x \in base}
licensee	=	{-case, -wh, ...}
licensor	=	{+case, -CASE, +wh, +WH, ...}

Label : *Leaves* \rightarrow select*(licensor)select*(base)licensee*P*I*

$$\mathcal{L}_{\mathcal{MG}} = \text{Closure}(\text{Lex}, F)$$



An Example MG: $G_{ww} = \langle NonSyn, Syn, Lex, F \rangle$

Example

Let G_{ww} be the MG with

$$Sem = \emptyset$$

$$Phon = \{1, 2\}$$

$$base = \{c, a_1, a_2, b, c_1, c_2, d\}$$

$$select = \{^=a_1, ^=a_2, ^=b, ^=c_1, ^=c_2, ^=d\}$$

$$Licensors = \{+L_1, +L_2\}$$

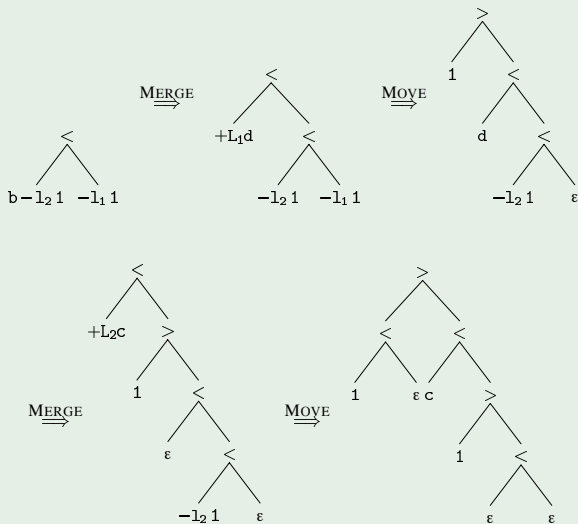
$$Licensees = \{-l_1, -l_2\}$$

$$Lex : i \in \{1, 2\}$$

$$\begin{array}{ll} \alpha_j = a_j - l_1 i & \gamma_i = ^=b + L_1 c_i - l_1 i & \zeta_1 = ^=b + L_1 d \\ \beta_j = ^=a_j b - l_2 i & \delta_i = ^=c_i + L_2 b - l_2 i & \zeta_2 = ^=d + L_2 c \end{array}$$

An Example Derivation

Example



Theorem (Harkema, Michaelis)

Minimalist Grammars (MG) in the sense of Stabler (1997) are weakly equivalent to multiple context-free grammars (MCFG) and are therefore mildly context-sensitive.

Idea of the Proof:

- define a finite partition
- each equivalence class is a set of trees such that the features triggering movement appear in identical structural positions
- each nonterminal in the MCFG represents such an equivalence class



Tree Version of the Harkema / Michaelis theorem:

Theorem (Mönnich)

Minimalist Grammars (Stabler 1997) are representable as multiple regular tree grammars (MREGT).

Idea of the Proof:

- Coding of tree structure by means of non-terminal symbols
- Non-terminal symbols ranging over tuples of subtrees



Multiple Context-Free Grammar (MCFG)

$$\mathcal{G} = \langle V_N, V_T, V_F, P, S \rangle$$

V_N is a ranked set of nonterminals

V_T is an unranked set of terminals

V_F are linear basic morphisms

$S \in V_N$ is the start symbol

P are the productions



Multiple Context-Free Grammar (MCFG)

$$A \longrightarrow f(A_0, \dots, A_{n-1})$$

$$A, A_0, \dots, A_{n-1} \in V_N$$

$$f \in V_F : (V_T^*)^k \longrightarrow (V_T^*)^l$$

$$k = \sum_{i=0}^{n-1} k_i \quad (k_i \text{ the rank of } A_i)$$

l the rank of A

Components of $(V_T^*)^l$ are strings over V_T and the components of $(V_T^*)^k$ and each component of $(V_T^*)^k$ occurs exactly once on the right-hand side.



Multiple Regular Tree Grammars (MREGT)

MREGT := Tree families generated by multiple regular tree grammars

$$A^{r_0} \rightarrow f(A_1^{r_1}, \dots, A_n^{r_n})$$

$$\begin{array}{ccc} \Delta & \dots & \Delta \\ t_1^1 & & t_{r_1}^1 \\ \vdots & & \vdots \\ \Delta & \dots & \Delta \\ t_1^n & & t_{r_n}^n \end{array} \Rightarrow (t_i^j \in T_\Sigma) \quad \Delta \quad \dots \quad \Delta (t'_k[t_i^j/x_i^j] = t_k, t_k \in T_\Sigma, t'_k \in T_{\Sigma \cup X})$$

Every x_i^j occurs exactly once in the v_0 -tuple (t'_1, \dots, t'_{r_0}) .



An Example Derivation

The multiple regular grammar G'_{ww} strongly equivalent to the minimalist example grammar G_{ww} uses the same operations with the decisive difference that the operations have to be relativized to the equivalence classes they apply to. The previous example derivation in term form

$$\text{move}(\text{merge}(\zeta_2, \text{move}(\text{merge}(\zeta_1, \text{merge}(\beta_1, \alpha_1))))))$$

has a similar structure in G'_{ww}

$$p(\text{move}_{W_2}(\text{merge}_{Z_2 X_1}(Z_2, \text{move}_{W_1}(\text{merge}_{Z,U}(Z_1, \text{merge}_{A_1 B_1}(B_1, A_1)))))))$$

where p serves to project tuples of trees onto their first component.



Top-Down Tree Transducer

Top-down tree transducers are finite state devices that transform in a strict recursive top-down manner an input tree into an output tree.

Definition (Top-Down Tree Transducer)

A **top-down tree transducer** is a tuple $\mathcal{T} = (Q, \Sigma, \Omega, q_0, R)$ where Σ and Ω are ranked alphabets, called the **input** and **output** alphabet, respectively, Q is a unary alphabet of **states**, q_0 is the **initial** state and R is a finite set of rules of the following form:

$$q(\sigma(x_1, \dots, x_m)) \rightarrow \xi$$

where $q \in Q$, $\sigma \in \Sigma_m$ and $\xi \in T_\Omega(\langle Q, X_m \rangle)$, i.e. a tree over $\Omega \cup \langle Q, X_m \rangle$ where each pair of a state q and a variable x_i is considered as an element of rank zero.

Theorem (Raoult)

$$MREGT = T_{fc}(REGT)$$

Corollary

$$MSOTT_{dir}(REGT) \supseteq MG$$



Tree Adjoining Grammars

- Tree-generating system
- Second-order substitution (adjunction)
- Factoring recursion from domain of dependency
- Extended domain of locality



Example

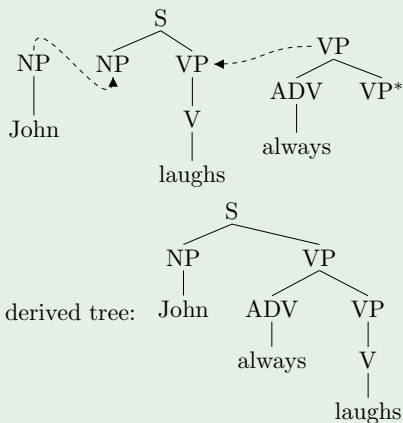


Figure : Sample TAG derivation (Kallmeyer 2009)

Definition (Tree Adjoining Grammar)

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



Definition

Let $G = \langle V_N, V_T, S, \mathcal{I}, \mathcal{A} \rangle$ be a TAG grammar. A **TAG derivation tree** is a tree over $\mathcal{I} \cup \mathcal{A}$ such that its root node is labeled by an initial tree with an S -root and all other nodes are labeled by auxiliary or initial trees in case of adjunction or substitution, respectively. A tree address is associated with each node indicating to which node in its parent tree the adjunction or substitution is applied.



Definition

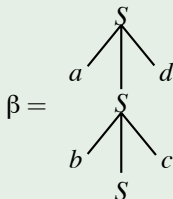
TAG derived trees are built from TAG derivation trees via **adjunction** or **substitution**. Adjunction is defined such that an auxiliary tree is spliced into a parent tree such that it basically “expands” an internal nonterminal. A subtree rooted in the node labelled with a nonterminal A is taken out of the a tree. A new auxiliary tree with both its root and a distinguished leaf node, the so called **foot node**, labeled with the same nonterminal is inserted in its place and the original subtree is appended at the foot node. Substitution on nonterminal leaf nodes, except for foot nodes, replaces this node by the substituted tree.



An example for a TAG generating the non-CF language $a^n b^n c^n d^n$ is given below:

Example

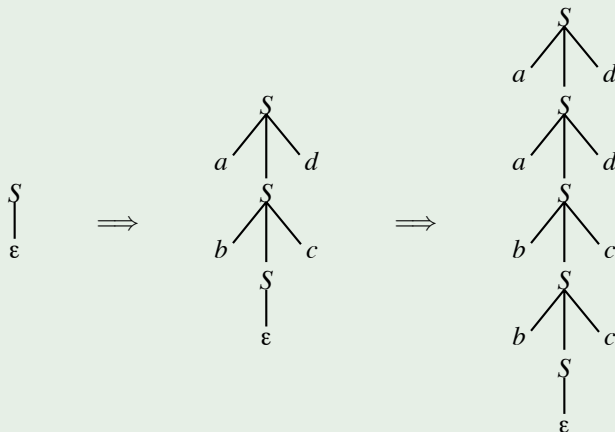
Let $G_{TAG} = \langle \{S\}, \{a, b, c, d\}, S, \{\alpha\}, \{\beta\} \rangle$ be a TAG. The only initial tree α and the only auxiliary tree β are given as follows:



Sample Derivation

A derivation with derivation tree $(\alpha(\beta(\beta)))$ (tree addresses omitted) yielding $abbccdd$ has only two steps, both adjoining the auxiliary tree in the same position:

Example



Extended TAG

- Introduced by Rogers (1998)
- Generalized variant of TAG
- Release of identity condition on labels of root and foot node in auxiliary trees
- Release of identity condition on labels of substituent node and root node of substituent auxiliary tree
- Similarity with subclass of Context-Free Tree Grammars



- **(Fujiyoshi and Kasai, Mönnich)** The class of string languages generated by monadic context-free tree grammars coincides with the class of string languages generated by *TAGs*.
- **(Kepser and Rogers)** The class of tree languages definable by extended *TAGs* is exactly the class of tree languages definable by monadic simple *CFTs*.
- **(Engelfriet and Maneth)** The class of tree languages generated by simple *CFTs* is exactly the class of output languages of simple macro tree transducers applied to the class of regular tree languages, i.e., $MTT_{si,sp}(REGT)$.



Definition

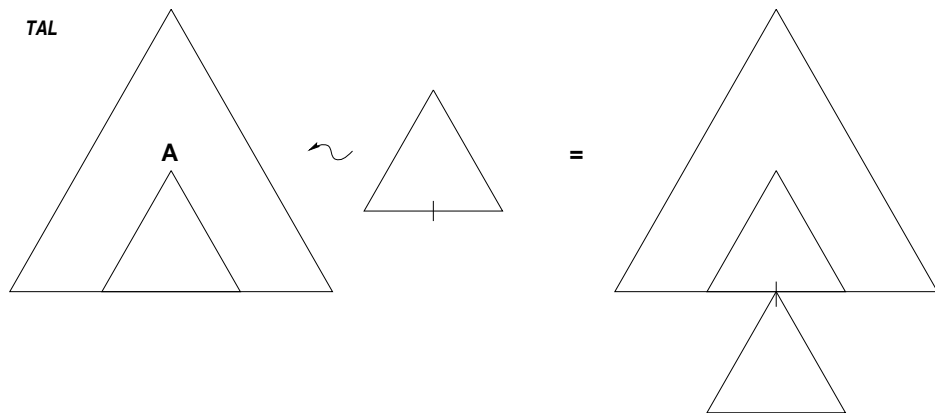
A *context-free tree (CFT)* grammar is a tuple $G = (\mathcal{F}, \Omega, S, P)$ where \mathcal{F} and Ω are ranked alphabets of non-terminals and terminals, respectively, $S \in \mathcal{F}^{(0)}$ is the start symbol and P is a finite set of productions of the form

$$F(y_1, \dots, y_m) \rightarrow \xi$$

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} . The class of grammars where all the non-terminals in a simple context-free tree grammar are at most of arity 1 is called **monadic simple** context-free grammars $CFT_{mon,sp}$.

Monadic Context-Free Tree Languages



Example

Consider the $C_{mon,sp}$ $G = \langle \{S, S', \bar{S}, E, \bar{a}, \bar{b}, \bar{c}, \bar{d}\}, \{a, b, c, d, \varepsilon, S_t, S_t^0\}, S', P \rangle$ with P given as follows:

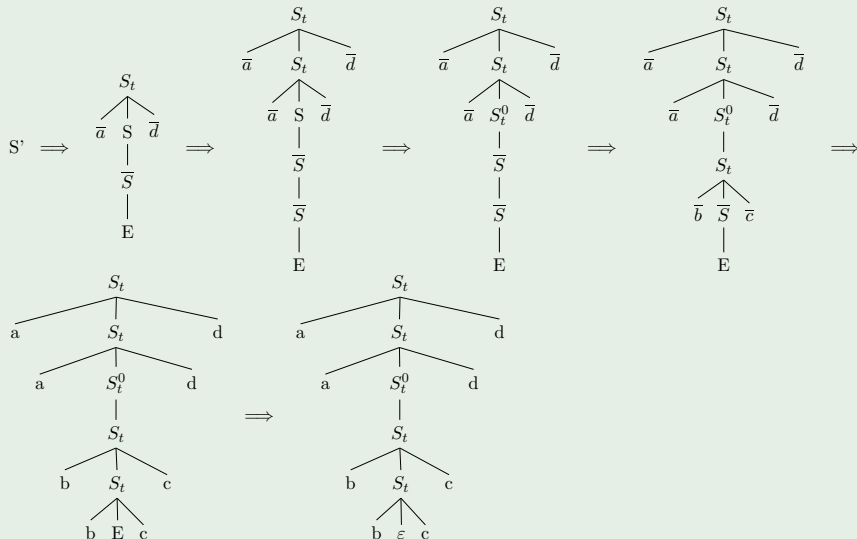
$$\begin{array}{ll} S' \longrightarrow S_t(\bar{a}, S(\bar{S}(E)), \bar{d}) & \bar{a} \longrightarrow a \\ S(y) \longrightarrow S_t(\bar{a}, S(\bar{S}(y)), \bar{d}) & \bar{b} \longrightarrow b \\ S(y) \longrightarrow S_t^0(y) & \bar{c} \longrightarrow c \\ \bar{S}(y) \longrightarrow S_t(\bar{b}, y, \bar{c}) & \bar{d} \longrightarrow d \\ & E \longrightarrow \varepsilon \end{array}$$

This grammar generates the language $L = \{a^n b^n c^n d^n\}$.



Example Derivation of $aabbccdd$

Example



Inspecting the rules of the example grammar it turns out that their right-hand sides exhibit a particular normal form with a terminal symbol as head and a possibly empty string of non-terminals. Context-free tree grammars of this type are in **Greibach** normal form.

Lemma

For any monadic context-free tree grammar G , there is a monadic context-free tree grammar G' in Greibach normal form such that

$$L(G') = L(G)$$



Attributed tree transducers are a variant of attribute grammars in which all attribute values are trees. Like macro tree transducers, attributed tree transducers can handle context information. However, in this model of tree transduction context information is treated in an explicit way. 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

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 :

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

where $\xi \in T_{\Omega \cup out_\sigma}$ and out_σ is the set of outside attributes of σ . Rules where ξ is (γ', ρ') are called **copy rules**.



Definition

For every $\sigma \in \Sigma_m$, the set of **inside attributes** is the set

$$ins_\sigma = \{(\alpha, \pi) \mid \alpha \in Syn\} \cup \{(\beta, \pi i) \mid \beta \in Inh, i \leq m\}$$

and the set of **outside attributes** is the set

$$out_\sigma = \{(\beta, \pi) \mid \beta \in Inh\} \cup \{(\alpha, \pi i) \mid \alpha \in Syn, i \leq m\},$$

where 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 $ATT_{ss,si}$.



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 $(\alpha', \pi') \rightarrow \xi$ and (α, π) labels one of the leaves in ξ . In **semantic** dependency graphs the direction of edges is reversed.

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



Definition

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

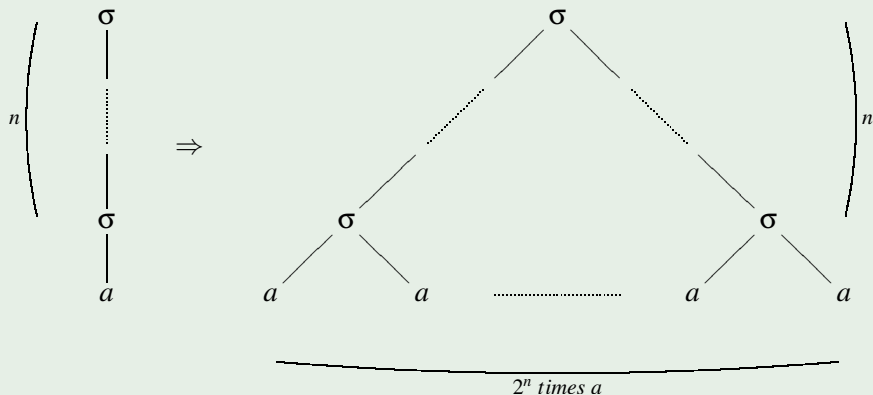
$$\tau_A = \{(s, t) \mid s \in T_\sigma, t \in T_\Omega, t = dec_s(\alpha_m, \epsilon)\}$$



Example *ATT* (Bloem, Engelfriet)

Example

Example We give an example of an attributed tree transducer with synthesized attributes that transform monadic trees into full binary trees:



Example

$$A = (\{\sigma^{(1)}, a^{(0)}\}, \{\sigma^{(2)}, a^{(0)}\}, \{\alpha\}, \alpha, R)$$

$R(\sigma)$:

$$(\alpha, \pi) \rightarrow \sigma((\alpha, \pi 1), (\alpha, \pi 1))$$

$R(a)$:

$$(\alpha, \pi) \rightarrow a$$

Remark: This transduction cannot be performed by a **simple** *ATT* since the size of an output tree is not linear in its input tree.



Lemma

For every $CFT_{mon,sp}$ G there is (a slight modification of) an equivalent non-copying reduced $MSOTT_{dir,rev}$ T .

Waypoints of Proof

- Translation of CFT G into equivalent ATT A
- Simulation of ATT A by $MSOTT$ T
- Pruning of T



Translation of *CFT G* into equivalent *ATT A*

The usual construction of an attributed tree transducer that outputs the same tree language that is generated by a given simple context-free tree grammar proceeds as follows: as input one takes the **derivation trees**, each input symbol gets assigned a single synthesized attribute and the number of inherited attributes corresponds to the arity of the non-terminal that is rewritten by the rule labeling the input symbol. The rules of the inside attributes simulate the **second-order substitution** of the original rewrite rules.



Translation of *CFT G* into equivalent *ATT A I*

For a given $CFG_{mon,sp} G = (\mathcal{F}, \Omega, S, P)$ an $1S, 1I - ATT_{ss,si} A_G = (Syn, Inh, \Sigma, \Omega, \alpha_m, R)$ with one synthesized and one inherited attribute only that outputs the same language is defined from the rules of G in the following way.

- $Syn = \{0, 1\}$ with $\alpha = \alpha_m$ at the root node
- $Inh = \{y\}$
- Every symbol in the derivation trees is assigned one synthesized attribute.
- If $p : N \rightarrow \xi$ is an element of P then R'_p is specified for both the synthesized and inherited attributes by structural induction on the right-hand side ξ :

$$(q, \pi) \rightarrow \vartheta(\xi),$$



Translation of *CFT G* into equivalent *ATT A II*

where $q = 0$ or $q = 1$ depending on the arity of N and ϑ substitutes a non-terminal M in ξ by (q, π_i) if the non-terminal M occurs in the i -th non-terminal position in ξ and y by (y, π)

$$(y_j, \pi_i) \rightarrow \vartheta'(\xi'),$$

where ξ' occurs in the argument position of some non-terminal L in ξ that itself occupies the i -th non-terminal position on the right-hand side of p and ϑ' is identical with ϑ except for erasing every y in ξ' .



Example

Applying the construction just outlined to the context-free tree grammar of the last example we obtain the following attributed tree transducer

$A = (Syn, Inh, \Sigma, \Omega, q_0, R')$:

- $Syn = \{0, 1\}$
- $Inh = \{y\}$
- $\Sigma = \{p_0, \dots, p_9\}$
- $\Omega = \{a, b, c, d, \varepsilon, S_t, S_t^0\}$
- $q_0 = 0$
- $R' = \bigcup_{p_i} R'_{p_i}$



Example ATT II

Example

$$R'_{p_1} = \{(0, \pi) \rightarrow S_t((0, \pi_1), (1, \pi_2), (0, \pi_4)), \\ (y, \pi_2) \rightarrow (1, \pi_3)\}$$

$$R'_{p_2} = \{(1, \pi) \rightarrow S_t((0, \pi_1), (1, \pi_2), (0, \pi_4)), \\ (y, \pi_2) \rightarrow (1, \pi_3), \\ (y, \pi_3) \rightarrow (y, \pi)\}$$

$$R'_{p_3} = \{(1, \pi) \rightarrow S_t^0((y, \pi))\}$$

$$R'_{p_4} = \{|1, \pi) \rightarrow S_t((0, \pi_1), (y, \pi)(0, \pi_2))\}$$

$$R'_{p_5} = \{(0, \pi) \rightarrow a\}$$

$$R'_{p_6} = \{(0, \pi) \rightarrow b\}$$

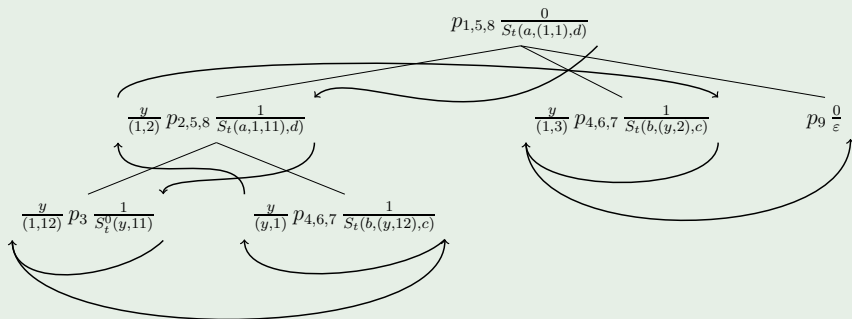
$$R'_{p_7} = \{(0, \pi) \rightarrow c\}$$

$$R'_{p_8} = \{(0, \pi) \rightarrow d\}$$

$$R'_{p_9} = \{(1, \pi) \rightarrow \varepsilon\}$$

Attributed Example Tree with Semantic Dependencies

Example



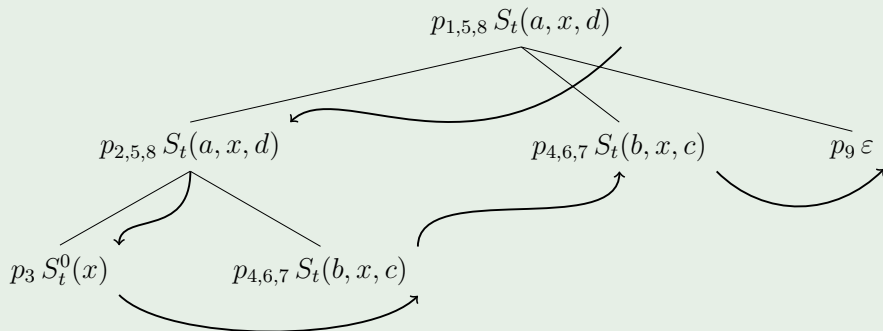
Simulation of *ATT* by *MSOTT*

Based on this intermediate simple *ATT* we can now define an equivalent *2-copy MSO* tree transducer T by specifying the edge formulas $\chi_{1,\gamma,\gamma'}(x,y)$ that represent the dependencies between the attributes and the node formulas $\psi_{d,\gamma}(x)$ that define the labels of the output tree. This transducer fulfills the condition of being **direction preserving/reversing** apart from the information flow to and fro between the the synthetic and inherited copies of the input trees. Bloem and Engelfriet show how to **prune** all occurrences of these transitions. Since this makes the "inherited" copy of the input tree superfluous we can erase it completely and arrive thus at a **slight modification** of a non-copying **reduced MSO**-transduction that is, furthermore, **direction preserving/reversing**.



Example

Applying the construction just outlined to the information transport illustrated by the example attributed dependencies we obtain the defined edges in the output of an equivalent *MSO*-transduction.



Lemma

For every (slight modification of a) non-copying reduced $MSOTT_{dir,rev} T$ there is an equivalent $CFT_{mon,sp} G_T$.

Waypoints of Proof

- Localization of $MSOTT$
- Relabeling of localized $MSOTT$
- Translation of relabeled $MSOTT$ into equivalent ATT
- Translation of ATT into equivalent CFT



Localization is required because there may be non-local upward edge definitions from the leaf of the input tree to a sibling of the first daughter of a dominating branching node. These non-local specifications can be localized by defining a local path on an "inherited" copy of the input tree that connects the endpoints of the original non-local upward edge.

Relabeling is needed because different occurrences of the same label in the (localized) input tree of the *MSO* translation may be differently connected to their copies, mothers, daughters and/or siblings. The specification of the equivalent *ATT* has to rely on information about the the (local) neighbourhood configurations established by the edge formulas $\chi_{i,c,d}(x,y)$. This information can be stored in the node labels by means of a relabeling *MSOTT*.



Proof idea (**Composition of attribute rules**)

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 second-order substitution or **adjunction**, for that matter, characteristic of context-free grammars.



Lemma

For every $1I, 1S - ATT_{ss,si}$ A , there exists an equivalent $CFT_{mon,sp}$ G_A .

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

- $\mathcal{F} = \Sigma$ where the arity of non-terminals is either zero or one depending on the occurrence of an inherited attribute assigned to them in the input tree.
- $\Omega = \Omega$
- $S = \{\sigma^{(0)}\}$ with $\sigma \in \Sigma$ labeling the root of an input tree.



From $1I, 1S - ATT_{ss,si}$ to $CFT_{mon,sp}$ II

- For every $\sigma \in \Sigma$ we construct a rule

$$\sigma(y_1, \dots, y_n) \rightarrow t$$

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

$$COMP(\xi) = \rho(t)$$

where ρ labels the i th daughter of σ and

$$t = COMP(rhs(\beta\pi i, \sigma))$$

- (ii) If $\xi = \beta\pi$ then

$$COMP(\xi) = y$$



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

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

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 T .

⊣



Theorem

The monadic simple context-free tree languages are exactly the output languages (of a slight modification) of non-copying direction preserving/reversing MSO definable tree transductions.



Set-Local Multicomponent Tree Adjoining Grammars

- Simultaneous adjunction of sequences of trees
- Motivated by linguistic applications
- Weakly equivalent with multiple context-free languages



Example

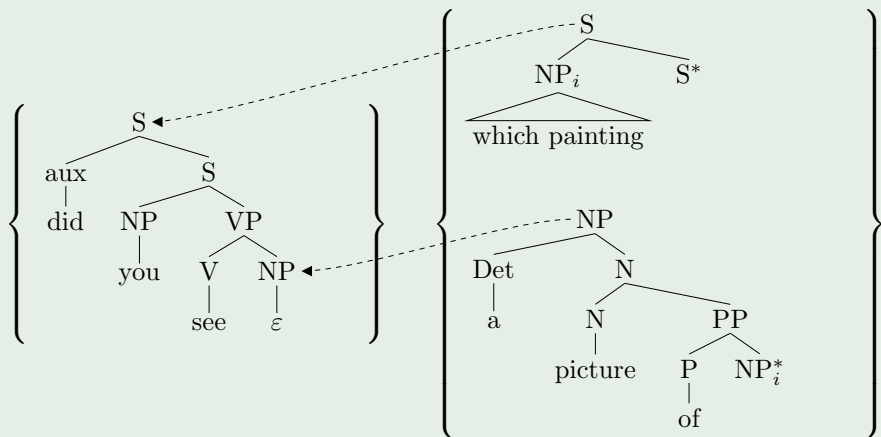


Figure : Sample MCTAG derivation (Kallmeyer 2009)

Definition (Set-Local Multicomponent Tree Adjoining Grammar)

A **Set-Local Multicomponent Tree Adjoining Grammar** (MCTAG) is a sextuple $\langle V_N, V_T, S, \mathcal{I}, \mathcal{A}, V_A \rangle$ where V_N is a finite set of **nonterminals**, V_T a finite set of **terminals**, $S \in V_N$ the **start symbol**, \mathcal{I} a finite set of **initial trees**, \mathcal{A} a finite set of **auxiliary trees** and V_A a finite set of sequences of auxiliary trees. Nonterminals stand for sequences of trees.



Definition (MCTAG Derivation Tree: Simultaneity Constraint)

Set-Local MCTA derivation trees are similar to simple TAG derivation trees, except for a simultaneity constraint induced by the notion of auxiliary tree sequences. For all components of an instance of an auxiliary tree sequence Γ there are components of another instance of an auxiliary tree sequence Γ' such that the components of Γ occur as adjunction daughters of components of Γ' .

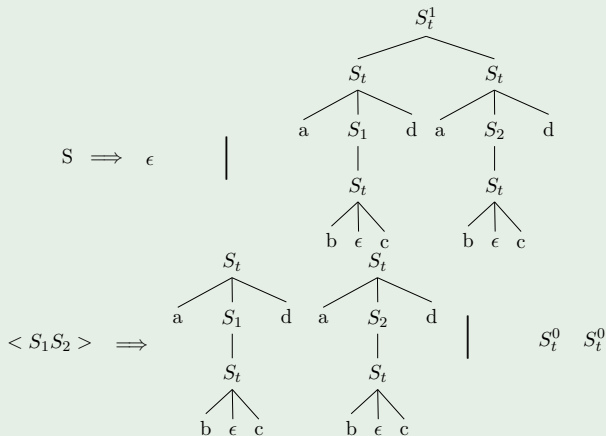


Example

Consider as an example the following Hyperedge Replacement Grammar $G = \langle N, \Sigma, S, P \rangle$ where $N = \{S, \langle S_1 S_2 \rangle\}$ is the set of linked nonterminal hyperedges with links $link(S) = (1)$ and $link(\langle S_1 S_2 \rangle) = (2, 2)$, $\Sigma = \{S_t, S_t^1, S_t^0, a, b, c, d\}$ is the set of terminals and P the set of rules of the two nonterminal hyperedges.



Example



Remarks

- 1 Set-local MCTAGs are a notational variant of (a special) form of tree generating hyperedge replacement grammars (*HR*)
- 2 The *HR* corresponding to a set-local MCTAG is in tree generating normal form with hyperforests as right-hand sides of its rules
- 3 Linked hyperedges assume the function of sequences of auxiliary trees
- 4 Replacement operation defined in the usual way without invoking simultaneity constraints



Definition

Given a finite ranked set Σ , a **hypergraph** g over Σ consists of finite sets of **nodes** V and **hyperedges** E . Every hyperedge of rank n (its **type**) is incident with n nodes and labeled by a symbol in $\Sigma^{(n)}$. The **type** of g is the number of the sequence of distinguished nodes assigned to g .



Definition

A context-free **hyperedge replacement grammar** HR is a tuple $G = \langle N, \Sigma, S, P \rangle$, where N is a ranked alphabet of **nonterminals**, Σ a ranked alphabet of **terminals** and P a finite set of **rules** of the form $A \Rightarrow g$, where $A \in N^{(k)}$, $k \geq 0$, and g is a hypergraph over $N \cup \Sigma$ of type k . The **language** $L(G)$ is generated by applying the rules in the expected way. If all elements of L are tree graphs the grammar G is called **tree-generating**. The family of tree languages generated by HR grammars is denoted by $TR(HR)$.



Definition

A **linked** ranked alphabet is a ranked alphabet Σ with a mapping *link* that assigns to every $\sigma \in \Sigma^{(n)}$ a sequence $link(\sigma) = (\rho_1, \dots, \rho_k)$ of natural numbers such that $\sum_{i=1}^k \rho_i = n$. A hypergraph g over Σ is a **linked forest** over Σ of **type** (ρ_1, \dots, ρ_k) if $cut(g) = h_1 \oplus \dots \oplus h_k$ for tree graphs h_1, \dots, h_k of type ρ_1, \dots, ρ_k , respectively, where $cut(g)$ is the result of replacing every hyperedge e in g with label σ and $link(\sigma) = (\rho_1, \dots, \rho_l)$ by distinct new edges e_1, \dots, e_l and where \oplus denotes disjoint union. A **tree graph** is a directed graph g in which each hyperedge e is incident with $n - 1$ **source** nodes and one **target** node and in which each node v is connected by a path to the unique **root** node.



Characterization of HR Tree Languages

The statement of the following theorem repeats the characterization of tree languages generated by HR grammars by means of the composition of finite-copying top-down tree transducers T_{fc} with macro tree transducers that are simple, i. e., linear and non-deleting in both their input and their parameters, $MTT_{si,sp}$:

Theorem (Drewes, Engelfriet, Maneth)

$$TR(HR) = T_{fc} \circ MTT_{si,sp}(REGT)$$

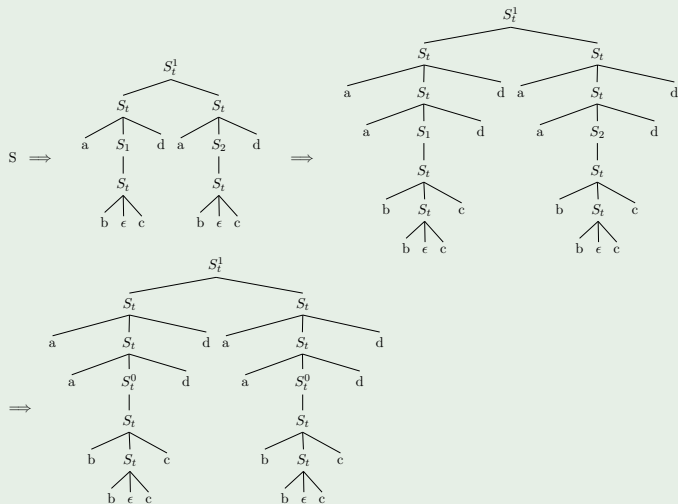
Since *MCTAGs* in tree generating (*HR*) normal form invoke only nonterminal hyperedges with *links* (1) or (2, ..., 2) the preceding characterization can be restricted to *monadic* simple macro tree transducers $MTT_{mon,si,sp}$ in the special case of multicomponent tree adjoining languages *MCTAL*:

Theorem

$$(Set - Local)MCTAL = T_{fc} \circ MTT_{mon,si,sp}(REGT)$$

Sample Derivation: *aabbccddaabbccdd*

Example



MCTAG Derivations and (simple) TAG Derivations I

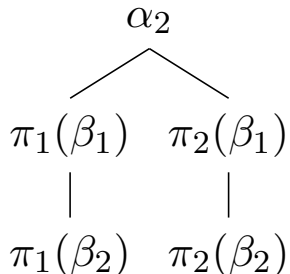
Denoting the right-hand sides of the rules for the nonterminal hyperedges S and $\langle S_1 S_2 \rangle$ by α_1, α_2 and β_1, β_2 , respectively, the preceding sequence of derived trees becomes the following *HR* derivation tree:

$$\begin{array}{c} \alpha_2 \\ | \\ \beta_1 \\ | \\ \beta_2 \end{array}$$



MCTAG Derivations and (simple) TAG Derivations II

The finite-copying top-down tree transducer T_{fc} turns this *HR* derivation tree into a familiar *TAG* derivation tree by decomposing the *HR* replacement steps into simultaneous steps of auxiliary *TAG* adjunctions:



Discussion

- Linked hyperedges correspond to states processing the same subtree
- Finite copying provides for the coordination of different subderivations
- Nodes in the output tree of the top-down transducer represent local adjunction operations
- Output trees of top-down transducer do not form a regular tree language
- Logical definition in terms of *MSO* transduction does not depend upon the satisfaction of simultaneity constraints by the input trees



Given the well-known inter-translatability between monadic simple macro tree transducers $MTT_{mon,si,sp}$ and simple attributed transducers with one synthesized and one inherited attribute only $1S, 1I - ATT_{ss,si}$ the automata-theoretic theorem for *MCTALs* can be restated as follows:

Theorem

$$(Set - Local)MCTAL = T_{fc} \circ 1S, 1I - ATT_{ss,si}(REGT)$$



Logical Definition of Set-Local MCTAGs I II

Our previous results regarding the relation between minimalist grammars MG , finite-copying top-down tree transducers T_{fc} and direction preserving tree to tree MSO transductions $MSOTT_{dir}$ on the one hand and between tree adjoining grammars TAG , simple attributed tree transducers with one synthesized and one inherited attribute only $1S, 1I - ATT_{ss,si}$ and non-copying direction preserving/reversing tree to tree transducers $MSOTT_{dir,rev}$ on the other lead to the logical version of the last theorem:

Theorem

$$MCTAL = MSOTT_{dir} \circ \text{non-copying } MSOTT_{dir,rev}(REGT)$$



- **Backwards Translation**

For every *MSO*-formula ϕ in the target structure \mathcal{Q} of an *MSO*-definable transduction $\Delta : \mathcal{R} \Rightarrow \mathcal{Q}$ one can construct an *MSO*-formula ϕ^Δ in \mathcal{R} such that

$$\Delta(\mathcal{R}) \models \phi \text{ iff } \mathcal{R} \models \phi^\Delta$$

- **Composition Closure**

The composition of two *MSO*-transductions is an *MSO*-transduction.

If T is a tree to tree transductions from T_Σ to T_Ω and

$T' = \langle \varphi', \nu', \psi', \chi' \rangle$ a tree to tree transduction from T_Ω to T_Δ then

their composition is specified by

$$T \circ T' = \langle (\varphi')^T, (\nu')^T, (\psi')^T, (\chi')^T \rangle.$$



Logical Definition of Set-Local MCTAGs II

Inspecting the definition of the component *MSO* formulae specifying the composition of two *MSO* tree to tree transductions we arrive at a second, more informative statement of the previous theorem. The logical characterization of (set-local) *MCTAGs* is basically the same as that of single-component *TAGs* except for the crucial difference that the direction preserving and the direction reversing paths defining the edges of the output structure are part of a **copying** *MSO* transduction:

Theorem

The set-local multicomponent tree adjoining languages are exactly the output languages (of a slight modification) of direction preserving/reversing MSO definable tree transductions.



- The leaf languages of set-local *MCTALs* cover the full family of $STR(MSOT(REGT))$, the output string languages of *MSO* definable transduction. The tree languages generated by *MCTAGs* are a proper subset of $TR(HR) = MSOTT(REGT)$.
- Open question whether the full spectrum of tree languages specifiable by means of *MSO* definable transductions is needed for the analysis of grammatical phenomena documented in natural languages.
- Related open problem: Extend the logical characterization of simple monadic context-free tree languages, equivalently of simple monadic macro tree transducers to a perspicuous configurational characterization of edge definitions in the larger class of simple context-free tree languages, equivalently in the class of simple finite-copying macro macro tree transducers.



Further Reading I



Bruno Courcelle.

The Monadic Second-order Logic of Graphs VII: Graphs as Relational Structures.

Theoretical Computer Science, 101:3–33, 1992.



Bruno Courcelle.

Graph Structure and Monadic Second-Order Logic. A Language-Theoretic Approach, volume 138 of *Encyclopedia of Mathematics and Its Applications*.

Cambridge University Press, 2012.






Frank Drewes.

A characterization of the sets of hypertrees generated by hyperedge-replacement grammars.

Theor. Comput. Sci., 32:159–208, 1999.







Further Reading II

-  Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel.
Hyperedge replacement graph grammars.
In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific Publishing, 1997.
-  Joost Engelfriet and Sebastian Maneth.
Macro Tree Transducers, Attribute Grammars, and MSO Definable Tree Translations.
Information and Computation, 154:34–91, 1999.
-  Joost Engelfriet and Sebastian Maneth.
Tree Languages Generated by Context-Free Graph Grammars.
In Hartmut Ehrig et al., editor, *Graph Transformation*, number 1764 in LNCS, pages 15–29. Springer, 2000.



Further Reading III

-  Joost Engelfriet and Grzegorz Rozenberg.
Node replacement graph grammars.
In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars*, pages 1–97. World Scientific Publishing, 1997.
-  Joost Engelfriet and Heiko Vogler.
Macro tree transducers.
Journal of Computer and System Sciences, 31(1):71–146, 1985.
-  Akio Fujiyoshi and Takumi Kasai.
Spinal-formed context-free tree grammars.
Theory of Computing Systems, 33(1):59–83, 2000.
-  Zoltán Fülöp and Heiko Vogler.
Syntax-Directed Semantics - Formal Models Based on Tree Transducers.
Springer, New York and Berlin, 1998.





Hendrik Harkema.

A characterization of minimalist grammars.

In P. et al. de Groote, editor, *Logical Aspects of Computational Linguistics (LACL '01)*, volume 2099 of *LNAI*, pages 193–211. Springer, Berlin, Heidelberg, 2001.



Aravind Joshi.

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, 1985.



Laura Kallmeyer.

A declarative characterization of different types of multicomponent tree adjoining grammars.

Research on Language and Computation, 7:55–99, 2009.





Laura Kallmeyer.

On mildly-context-sensitive non-linear rewriting.

Research on Language and Computation, 8:341–363, 2010.



Stephan Kepser and James Rogers.

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*, 2007.



Jens Michaelis.

Derivational minimalism is mildly context-sensitive.

In M. Moortgat, editor, *Logical Aspects of Computational Linguistics (LACL '98)*, volume 2014 of *LNAI*, pages 179–198, 1998.





Uwe Mönnich.

Adjunction as substitution.

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



Uwe Mönnich.

Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions.

In J. Rogers and S. Kepser, editors, *Proceedings Model Theoretic Syntax at 10*, 2007.



Uwe Mönnich.

Well-nested tree languages and attributed tree transducers.

In Srinivas Bangalore, Robert Frank, and Maribel Romero, editors, *Proceedings of the 10th International Workshop on Tree Adjoining Grammars and Related Formalisms TAG+11*, 2010.



Further Reading VII



Uwe Mönnich.

A logical characterization of extended TAGs.

In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 2012.



Jean-Claude Raoult.

Rational Tree Relations.

Bull. Belg. Math. Soc., 4:149–176, 1997.



William C. Rounds.

Mappings and grammars on trees.

Mathematical Systems Theory, 4(3):257–287, 1970.



Edward Stabler.

Computational perspectives on minimalism.

In C. Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–642. Oxford University Press, 2011.





Edward P. Stabler.

Derivational minimalism.

In C. Retoré, editor, *Logical Aspects of Computational Linguistics (LACL '96)*, volume 1328 of *LNAI*. Springer, Berlin, Heidelberg, 1997.



David J. Weir.

Characterizing Mildly Context-Sensitive Grammar Formalisms.

PhD thesis, University of Pennsylvania, Philadelphia, PA, 1988.

