A Formal Approach to Non-Context-Free Phenomena

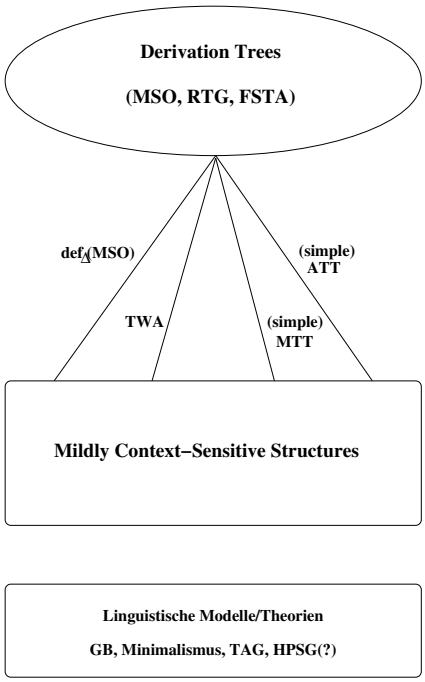
Uwe Mönnich
Frank Morawietz

Seminar für Sprachwissenschaft
Universität Tübingen

{um,frank}@sfs.nphil.uni-tuebingen.de

http://tcl.sfs.nphil.uni-tuebingen.de/

Passau, 24.10.2000



Outline

- Motivation and Theoretical Background
- Derivational Minimalism
- From Minimalism to Multiple Context-Free Grammars
- From MCFGs to Regular Tree Grammars
- Coding of the Intended Structures

Linguistic Background

J. Rogers (1994,98) gives a formalization within MSO tree logic of a main paradigm in Theoretical Linguistics:

The Principle and Parameters Approach

in the tradition of N. Chomsky

+ Natural Formalization of (almost) all Principles
+ Modular
+ Decidable
+ Descriptive Complexity Result
− Undecidability of *Free Indexation* (Grid)
− Non Context-Free Phenomena in Natural Language

Rogers (1994,1998): $\mathbf{L^2_{K,P}}$

- K a set of individual constants
- P a set of predicate constants
- immediate dominance: $\lhd$
- reflexive dominance: $\lhd^*$
- proper dominance: $\lhd^+$
- proper precedence: $\prec$
- usual connectives, variables, equality, set membership and quantifiers:
  - a set of *monadic* second-order variables (X,Y,...);
  - a set of individual variables (x,y,...);
  - boolean connectives: $\neg$, $\wedge$, ...;
  - Quantifiers over individuals and sets: $\exists$, $\forall$;
  - equality and membership: $\approx$, $\in$.

Linguistic Motivation

- Non-Contextfree Phenomena:

  (... 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

- Derivational Minimalism is Non-Contextfree

Formal Background

Weak Equivalence of Mildly Context-Sensitive Formalisms:

STR(HR): languages generated by string generating hyperedge replacement grammars

OUT(DTWT): output languages of deterministic tree-walking tree-to-string transducers

yDT$_{\mathbf{fc}}$(REGT): yields of images of the regular tree languages under deterministic finite-copying top-down tree transductions

MCFL: languages generated by multiple context-free grammars

MCTAL: languages generated by multi-component tree adjoining grammars

LCFRL: languages generated by linear context-free rewriting systems

LUSCL: languages generated by local unordered scattered context grammars

General Structure of the Approach

Translating Minimalist Grammars to MCFGs

$\Rightarrow$ Regular Characterization of the MCFG-Languages

$\Rightarrow$ Retrieval of the Intended Structures via Regular Relations

Minimalist Grammars (Stabler, 1997)

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

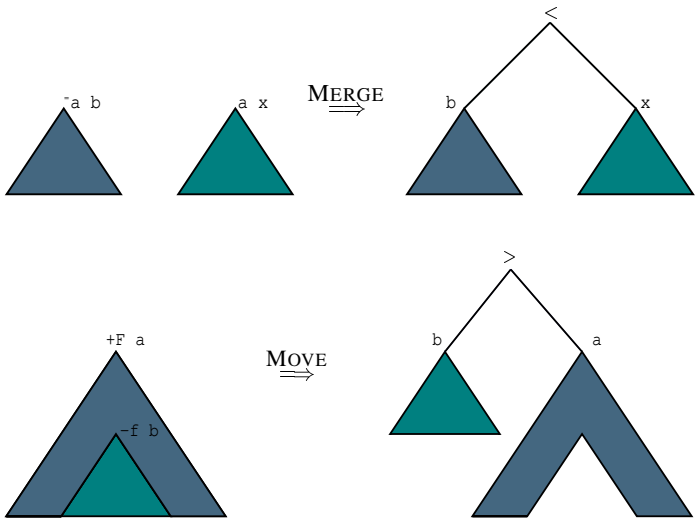
$NonSyn$	=	$Phon \cup Sem$
Syn	=	base $\cup$ select $\cup$ licensee $\cup$ licensor
Lex	=	a finite set of trees
F	=	$\{\text{MERGE}, \text{MOVE}\}$

$Phon$	are the phonological features
Sem	are the interpreted/semantical features
base	$= \{\texttt{c}, \texttt{t}, \texttt{v}, \texttt{d}, \texttt{n}, \dots\}$
select	$= \{\texttt{=x}, \texttt{=X}, \texttt{X=} \mid \texttt{x} \in \text{base}\}$
licensee	$= \{\texttt{-case}, \texttt{-wh}, \dots\}$
licensor	$= \{\texttt{+case}, \texttt{+CASE}, \texttt{+wh}, \texttt{+WH}, \dots\}$

$Label : Leaves \longrightarrow \text{select}^*(\text{licensor})\text{select}^*(\text{base})\text{licensee}^* P^* I^*$

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

MERGE/MOVE



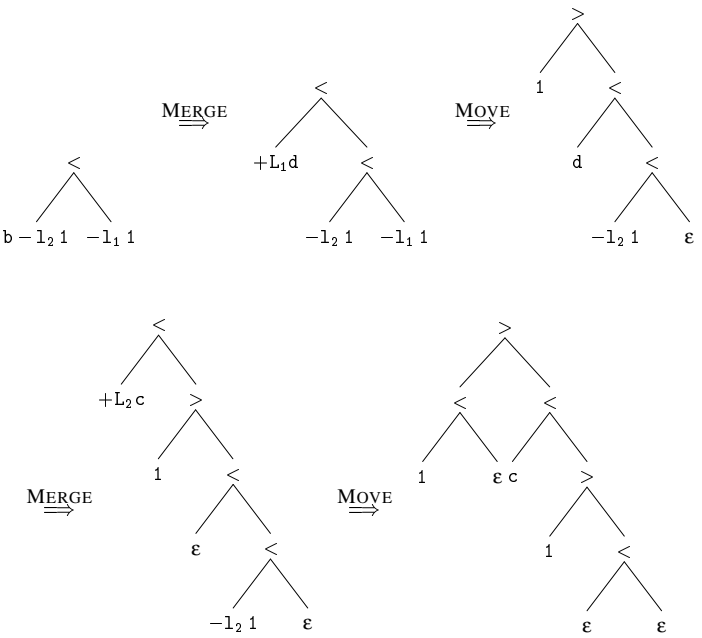
An Example MG: $G_{ww} = \langle NonSyn, Syn, Lex, F \rangle$

$$Sem = \emptyset$$
$$Phon = \{1, 2\}$$
$$base = \{\texttt{c}, \texttt{a}_1, \texttt{a}_2, \texttt{b}, \texttt{c}_1, \texttt{c}_2, \texttt{d}\}$$
$$select = \{\texttt{=a}_1, \texttt{=a}_2, \texttt{=b}, \texttt{=c}_1, \texttt{=c}_2, \texttt{=d}\}$$
$$Licensors = \{\texttt{+L}_1, \texttt{+L}_2\}$$
$$Licensees = \{\texttt{-l}_1, \texttt{-l}_2\}$$

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

α_i	=	$\texttt{a}_i\texttt{-l}_1 i$	γ_i	=	$\texttt{=b+L}_1\texttt{c}_i\texttt{-l}_1 i$	ζ_1	=	$\texttt{=b+L}_1\texttt{d}$
β_i	=	$\texttt{=a}_i\texttt{b-l}_2 i$	δ_i	=	$\texttt{=c}_i\texttt{+L}_2\texttt{b-l}_2 i$	ζ_2	=	$\texttt{=d+L}_2\texttt{c}$

An Example Derivation



A Linguistic Example: Naive Tamil (Stabler, 1999)

`lavinia::d`	`titus::d`
`vt praise`	`v criticize`
`v laugh`	`v cry`
`ε::=i d`	`-s::=pred +V i`
`ε::=vt =d =d pred -v`	`ε::=v =d pred -v`

MERGE/MOVE: Formal Definition

$$\frac{s::{=}f\gamma \qquad t\cdot f,\alpha_1,\ldots,\alpha_k}{st:\gamma,\alpha_1,\ldots,\alpha_k}\text{merge1}$$

$$\frac{s:{=}f\gamma,\alpha_1,\ldots,\alpha_k \qquad t\cdot f,\beta_1,\ldots,\beta_l}{ts:\gamma,\alpha_1,\ldots,\alpha_k,\beta_1,\ldots,\beta_l}\text{merge2}$$

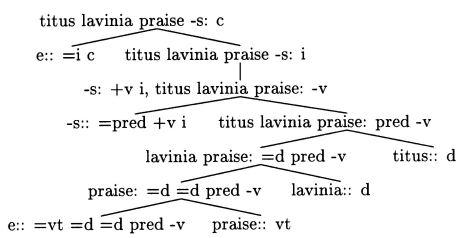
$$\frac{s\cdot {=}f\gamma,\alpha_1,\ldots,\alpha_k \qquad t\cdot f\delta,\beta_1,\ldots,\beta_l}{s:\gamma,\alpha_1,\ldots,\alpha_k,\ t:\delta,\beta_1,\ldots,\beta_l}\text{merge3}$$

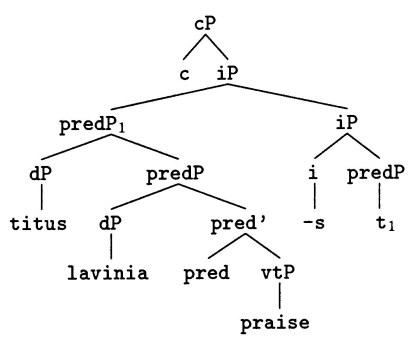
$$\frac{s:{+}f\gamma,\alpha_1,\ldots,\alpha_{i-1},\ t:{-}f,\alpha_{1+1},\ldots,\alpha_k}{ts:\gamma,\alpha_1,\ldots,\alpha_{i-1},\alpha_{i+1},\ldots,\alpha_k}\text{move1}$$

$$\frac{s:{+}f\gamma,\alpha_1,\ldots,\alpha_{i-1},\ t:{-}f\delta,\alpha_{i+1},\ldots,\alpha_k}{s:\gamma,\alpha_1,\ldots,\alpha_{i-1},\ t:\delta,\alpha_{i+1},\ldots,\alpha_k}\text{move2}$$

$s,t\in\Sigma^*$, $\cdot\in\{:,::\}$, $f\in\mathsf{base}$, $\gamma\in Syn^*$, $\delta\in Syn^+$, α_i,β_j, $0\le i,j\le k,l$ are chains.

A resulting tree





Derivational Minimalism is mildly context-sensitive

Theorem (Michaelis). Each Minimalist Grammar (MG) in the sense of Stabler (1997) can be translated into a weakly equivalent multiple context-free grammar (MCFG) and is therefore mildly context-sensitive.

Idea of the Proof:

- define a finite partition
- each euqivalence 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

MCFG

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

V_N is a ranked set of nonterminals

V_T is a ranked set of terminals

V_F are linear basic morphisms

S $\in V_N$ is the start symbol

P are the productions

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

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

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

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

l the rank of A

An Example Grammar

$$\mathcal{G} = \langle \{S, A\}, \{a_1, a_2, a_3\}, S, P \rangle$$

$$P = \left\{ \begin{array}{lcl} S & \rightarrow & g(A) \\ A & \rightarrow & f() \\ A & \rightarrow & h(A) \end{array} \right\}$$

$$\begin{array}{lcl} g(x,y,z) & = & xyz \\ f() & = & \langle a_1, a_2, a_3 \rangle \\ h(x,y,z) & = & \langle xa_1, ya_2, za_3 \rangle \end{array}$$

$$\begin{array}{lcl} a_1, a_2, a_3 & : & (0,1) \\ A & : & (0,3) \\ \bullet, \pi_1^3, \pi_2^3, \pi_3^3 & : & (3,1) \end{array}$$

$$\mathcal{L}(\mathcal{G}) = a_1^n a_2^n a_3^n$$

An example derivation

$$\begin{array}{l} S \\ \rightarrow g(A) \\ \rightarrow g(h(A)) \\ \rightarrow g(h(f())) \\ \rightarrow g(h(\langle a_1, a_2, a_3 \rangle)) \\ \rightarrow g(\langle a_1 a_1, a_2 a_2, a_3 a_3 \rangle) \\ \rightarrow a_1 a_1 a_2 a_2 a_3 a_3 \end{array}$$

Basic Definitions I: many-sorted signature

Let S be a set of sorts. A *many-sorted signature* Σ is an indexed family

$$\langle \Sigma_{w,s} \mid w \in S^*, s \in S \rangle$$

of disjoint sets. A symbol in $\Sigma_{w,s}$ is called an:

operator of type $\langle w, s \rangle$,

arity w,

sort s and

rank $|w|$.

In the case of a single-sorted signature we write $\Sigma_{s^n,s}$ as Σ_n.

Basic Definitions II: $T(\Sigma)$

- If $\sigma \in \Sigma_{\varepsilon,s}$ for some $s \in \mathcal{S}$ then σ is a (trivial) tree of sort s.
- If, for some $s \in \mathcal{S}$ and $w = s_1 \cdots s_n$ with $s_i \in \mathcal{S}$, $\sigma \in \Sigma_{w,s}$ and $t_1, \ldots, t_n \in T(\Sigma)$ with t_i of sort s_i then $\sigma(t_1, \ldots, t_n)$ is a tree of sort s.

Basic Definitions III: Σ-algebra

A Σ-*algebra* $\mathbb{A}$ consists of an S-indexed family of sets $A = \langle A^s \rangle_{s \in S}$ and for each operator $\sigma \in \Sigma_{w,s}$ a function

$$\sigma_{\mathbb{A}} : A^w \to A^s$$
$$A^w = A^{w_1} \times \cdots \times A^{w_n}$$
$$w = w_1 \cdots w_n.$$

The set $T(\Sigma)$ can be made into a Σ-algebra $\mathbb{T}$ by specifying the operations as follows.

For every $\sigma \in \Sigma_{w,s}$ and every $t_1, \ldots, t_n \in T(\Sigma)$ we identify

$$\sigma_{\mathbb{T}}(t_1, \ldots, t_n) \text{ with } \sigma(t_1, \ldots, t_n).$$

Lawvere Algebra I

Given a set of sorts S, a Lawvere theory, as an algebra, is an $S^* \times S^*$-sorted algebra $\mathbb{A}$, whose carriers $\langle A^{(u,v)} | u, v \in S^* \rangle$ consist of the morphisms of the theory and whose operations are of the following types ($n \in \mathbb{N}$, $u = u_1 \cdots u_n$ with $u_i \in \mathcal{S}$ for $1 \leq i \leq n$ and $v, w \in \mathcal{S}^*$):

projection:	$\pi_i^u \in A^{\langle u, u_i \rangle}$
composition:	$c_{(u,v,w)} \in A^{\langle u,v \rangle} \times A^{\langle v,w \rangle} \to A^{\langle u,w \rangle}$
target tupling:	$(\quad)_{(v,u)} \in A^{\langle v,u_1 \rangle} \times \cdots \times A^{\langle v,u_n \rangle} \to A^{\langle v,u \rangle}$

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

Basic Definitions IV: $T(\Sigma, X)$

Let Σ be a single-sorted signature and $X = \{x_1, x_2, x_3, \ldots\}$ a countable set of variables. For $k \in \mathbb{N}$ define $X_k \subseteq X$ as $\{x_1, \ldots, x_k\}$.

- $T(\Sigma, X_k)$ (over Σ) is the set of trees $T(\Sigma')$ over the single-sorted signature $\Sigma' = \langle \Sigma'_n \,|\, n \in \mathbb{N} \rangle$, where $\Sigma'_0 = \Sigma_0 \cup X_k$ and $\Sigma'_n = \Sigma_n$ for $n > 0$.
- $T(\Sigma, X_k) \subseteq T(\Sigma, X_l)$ for $k \leq l$.
- $T(\Sigma, X) = \bigcup_{k \in \mathbb{N}} T(\Sigma, X_k)$.

Lawvere Algebra II

With S being a singleton, the power set $\wp(T(\Sigma, X_k))$ of k-ary trees constitutes the central example of interest for formal language theory. The carriers

$$\langle \wp(T(k,m)) \,|\, k,m \in w \rangle$$

consist of 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 $\{(t_1,\ldots,t_m)\}$.

Regular Tree Grammars

$$\Gamma = \langle \Sigma, \mathsf{F}_0, S, \mathsf{P} \rangle \text{ where}$$

$$\Sigma = \langle \Sigma_{w,s} \,|\, w \in \mathcal{S}^*, s \in \mathcal{S} \rangle$$

$$\mathsf{F}_0 = \langle \mathsf{F}_{\varepsilon,s} \,|\, s \in \mathcal{S} \rangle$$

$S \in \mathsf{F}_0$ is the starting symbol

P is a set of productions

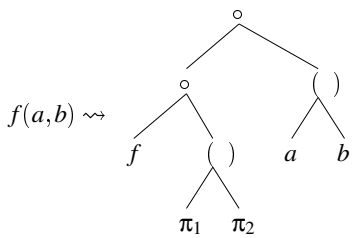
The $p \in \mathsf{P}$ have the form

$$F \longrightarrow t$$

with $F \in \mathsf{F}_{\varepsilon,s}$, and t a term over $T(\Sigma \cup \mathsf{F}_0)$.

An application of a rule $F \longrightarrow t$ "rewrites" a tree rooted in F as the tree t.

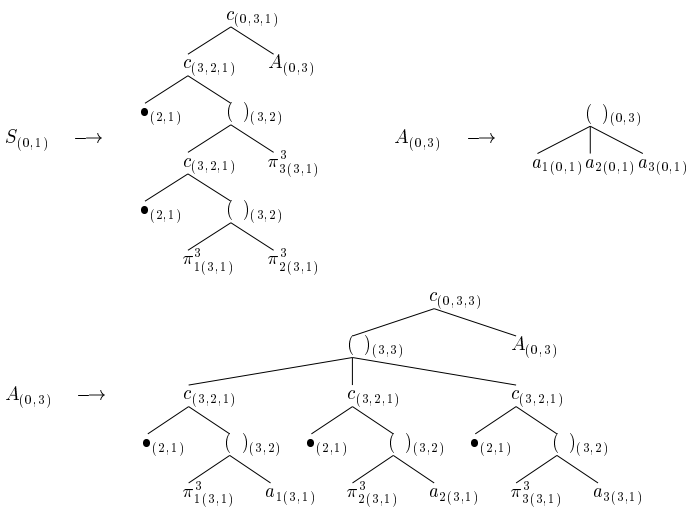
From MCFGs to RTGs: Intuition

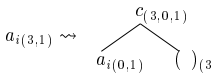


$$X \to f(Y) \;\; \rightsquigarrow \;\; X_{(j,l)} \to c_{(j,k,l)}(\mathsf{T}(f(x_1,\ldots,x_k)), Y_{(j,k)})$$

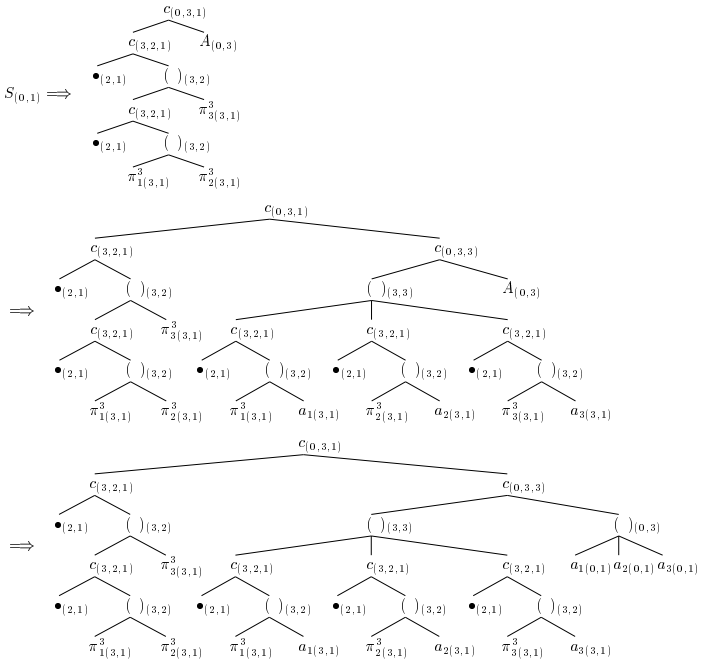
f is a mapping from k-tuples to l-tuples of terminal strings.

The Translated Example Grammar $\mathcal{G}'$





An example derivation of $\mathcal{G}'$



Coding RTGs in MSO Logic

- Define a tree automaton $\mathfrak{A}_{\mathcal{G}'}$ from the RTG $\mathcal{G}'$
- Code its behaviour in $\varphi_{\mathfrak{A}_{\mathcal{G}'}}$

$$(\exists X_0,\ldots,X_m)\Big[\bigwedge_{i\neq j}(\neg\exists y)[y\in X_i\wedge y\in X_j]\wedge$$
$$(\forall x)[(\neg\exists y)[x\lhd y]\rightarrow x\in X_0]\wedge$$
$$(\forall x_1,\ldots,x_n,y)\Big[\bigvee_{\substack{(i_1,\ldots,i_n,\sigma,j)\in\alpha\\ 1\leq k\leq n}} x_k\in X_{i_k}\wedge y\lhd x_k\wedge y\in X_j\wedge y\in P_\sigma\Big]$$
$$\bigvee_{i\in F}(\exists x\forall y)[x\lhd^* y\wedge x\in X_i]$$

Definition (Bottom-Up Tree Automaton).

$$\mathfrak{A} = \langle A, \Sigma, \delta, a_0, F\rangle$$

- A the (finite) set of states,
- Σ a ranked alphabet,
- $a_0 \in A$ the initial state,
- $F \subseteq A$ the final states and
- $\delta : \bigcup_n (A^n \times \Sigma_n) \rightarrow A$ the transition function.

$$h_\delta(\varepsilon) = a_0$$
$$h_\delta(\sigma(t_1,\ldots,t_n)) = \delta(h_\delta(t_1),\ldots,h_\delta(t_n),\sigma)$$
$$t_i \in T_\Sigma, 1\leq i\leq n, \sigma\in\Sigma_n$$

$$T(\mathfrak{A}) = \{t \mid h_\delta(t)\in F\}$$

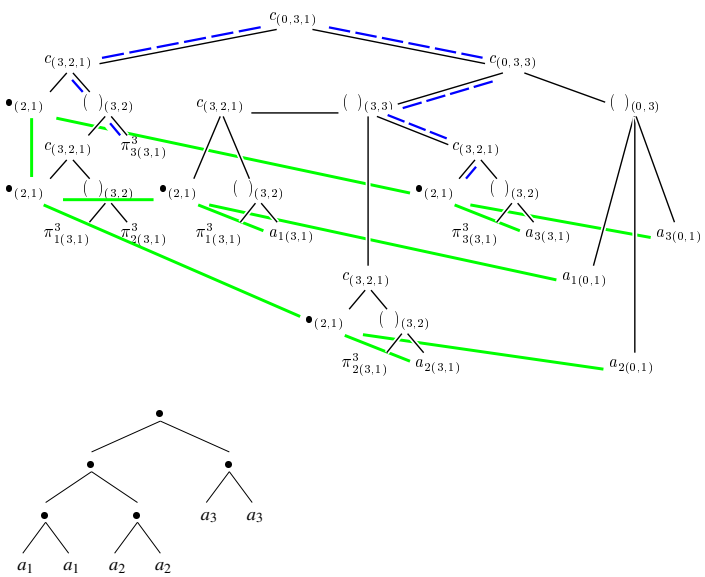
Goal:

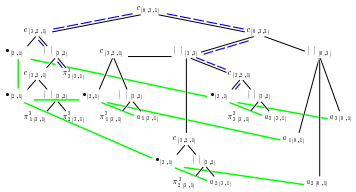
Define a set of relations
$R^I = \{\blacktriangleleft^\star$ *(dominance)*, $\blacktriangleleft, \blacktriangleleft^+, \lhd$ *(precedence)*, *c-command*, $\ldots\}$
holding between the nodes $n \in N^L$ of the explicit tree T^L
which carry a *"linguistic"* label $l \in \mathcal{T}^L \cup \mathcal{N}^L\,(\subseteq \bigcup_{n\geq 0}\Sigma_n)$
in such a way, that when interpreting

$\blacktriangleleft^\star \in R^I$ as a tree order on the set of *"linguistic"* nodes and
$\lhd \in R^I$ as the precedence relation on the resulting structure,

$\langle\{n \mid n\in N^L \wedge \ell(n)\in\mathcal{T}^L\cup\mathcal{N}^L\}, \blacktriangleleft^\star, \lhd\rangle$
is in fact the intended tree corresponding to T^L.

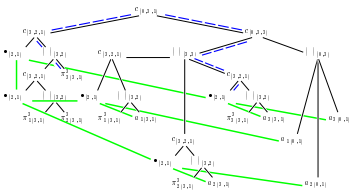
Reconstructing the Intended Structures





A few general facts

1. Our trees feature three families of labels: the "linguistic" symbols L, i.e., the lifted symbols of the underlying MCFG; the "composition" symbols $C = \{c_{(u,v,w)}\}$; the "tupling" symbols $(\quad)_{(v,u)}$ and the "projection" symbols $\Pi = \{\pi_i^k\}$.
2. All nonterminal nodes in t' are labeled by some $c \in C$ 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.



A few facts (cont'd)

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 $c \in C$ 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 $c \in C$) which properly dominates ν and which properly dominates a node in its other subtree 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 reachable from a leftmost daughter and bears a composition symbol. This crucial fact is arrived at by induction on the construction of $\mathcal{G}'_{ww}$ from $\mathcal{G}_{ww}$.

MSO definable transductions: $\mathcal{R} \rightsquigarrow \mathcal{Q}$

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

φ the domain of the transduction

ψ the resulting domain of $\mathcal{Q}$

θ_q the new relations

Definition (Tree-Walking Automaton). 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 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)",
- $\varphi(x)$ "verify that φ holds at the current node".

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

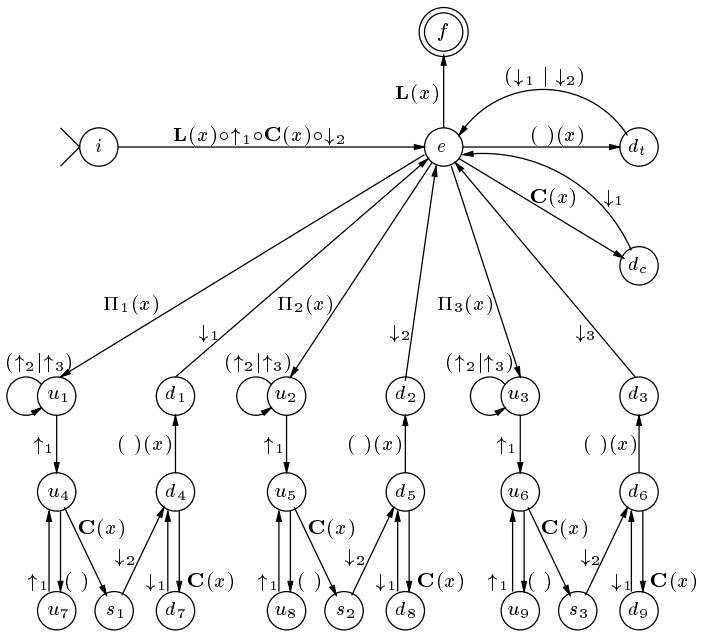
$$R_t(\mathfrak{A}) = \{(x,y) \mid (x,q_i) \stackrel{*}{\Longrightarrow} (y,q_f) \text{ for some } q_i \in I \text{ and some } q_f \in F\}$$

where for all states $q_i, q_j \in Q$ and nodes x, y in t

$$(x,q_i) \Rightarrow (y,q_j) \text{ iff } \exists d \in \Delta : (q_i, d, q_j) \in \delta \text{ and } y \text{ is reachable from } x \text{ in } t \text{ via } d.$$

If all the tests $\varphi(x)$ of $\mathfrak{A}$ are *MSO* definable, $\mathfrak{A}$ specifies a *regular tree-node relation*, which is itself *MSO* definable (B & E 1997).

The tree-walking automaton $A_{\blacktriangleleft}$



The walking language $W_{\blacktriangleleft}$

$$W_{\blacktriangleleft} = \mathrm{L}(x) \cdot \uparrow_1 \cdot \mathrm{C}(x) \cdot \downarrow_2 \cdot (W_{(\ )} \cup W_{\mathrm{C}} \cup W_{\Pi_1} \cup W_{\Pi_2} \cup W_{\Pi_3})^* \cdot \mathrm{L}(x)$$

$$\begin{aligned}
W_{(\ )} &= (\ )(x) \cdot (\downarrow_1 \cup \downarrow_2) \\
W_{\mathrm{C}} &= \mathrm{C}(x) \cdot \downarrow_1 \\
W_{\Pi_1} &= \Pi_1(x) \cdot (\uparrow_2 \cup \uparrow_3)^* \cdot \uparrow_1 \cdot ((\ )(x) \cdot \uparrow_1)^* \cdot \mathrm{C}(x) \cdot \downarrow_2 \cdot (\mathrm{C}(x) \cdot \downarrow_1)^* \cdot (\ )(x) \cdot \downarrow_1 \\
W_{\Pi_2} &= \Pi_2(x) \cdot (\uparrow_2 \cup \uparrow_3)^* \cdot \uparrow_1 \cdot ((\ )(x) \cdot \uparrow_1)^* \cdot \mathrm{C}(x) \cdot \downarrow_2 \cdot (\mathrm{C}(x) \cdot \downarrow_1)^* \cdot (\ )(x) \cdot \downarrow_2 \\
W_{\Pi_3} &= \Pi_3(x) \cdot (\uparrow_2 \cup \uparrow_3)^* \cdot \uparrow_1 \cdot ((\ )(x) \cdot \uparrow_1)^* \cdot \mathrm{C}(x) \cdot \downarrow_2 \cdot (\mathrm{C}(x) \cdot \downarrow_1)^* \cdot (\ )(x) \cdot \downarrow_3
\end{aligned}$$

$W_{\blacktriangleleft}$ can be inductively translated into a *MSO*-formula

$$trans_{W_{\blacktriangleleft}}(x,y).$$

Reflexive transitive closure in *MSO* (on finite structures)

$$\textit{R-closed}(X) \equiv (\forall x,y)[x \in X \wedge R(x,y) \rightarrow y \in X]$$

$$[R^*(x,y) \equiv (\forall X)[\textit{R-closed}(X) \wedge x \in X \rightarrow y \in X]$$

Walking language $\Rightarrow$ *MSO*-formula

$$
\begin{aligned}
trans_{\emptyset}(x,y) &= \bot \\
trans_{\downarrow_i}(x,y) &= edge_i(x,y) \\
trans_{\uparrow_i}(x,y) &= edge_i(y,x) \\
trans_{\varphi(x)}(x,y) &= \varphi(x) \wedge x = y \\
trans_{W_1 \cup W_2}(x,y) &= trans_{W_1}(x,y) \vee trans_{W_2}(x,y) \\
trans_{W_1 \cdot W_2}(x,y) &= \exists z\,(trans_{W_1}(x,z) \wedge trans_{W_2}(z,y)) \\
trans_{W^*}(x,y) &= trans^*_W(x,y) \\
trans^*_W(x,y) &= \forall X\,(\forall v,w\,(v \in X \wedge trans_W(v,w) \;\rightarrow\; w \in X) \\
&\qquad \wedge x \in X \;\rightarrow\; y \in X)
\end{aligned}
$$

Intended Structures via MSO transduction

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

$$Q = \{\blacktriangleleft, \blacktriangleleft^\star, \blacktriangleleft^+, \lhd, \dots\}$$

$$
\begin{aligned}
\varphi &= \text{ the formula coding } \mathfrak{A}_{G'} \\
\psi &= (\exists y)[\mathsf{trans}_{W_\blacktriangleleft}(x,y) \vee \mathsf{trans}_{W_\blacktriangleleft}(y,x)] \\
\theta_\blacktriangleleft &= \mathsf{trans}_{W_\blacktriangleleft}(x,y) \\
\theta_{\blacktriangleleft^\star} &= (\forall X)[\blacktriangleleft\text{-}closed(X) \vee x \in X \;\rightarrow\; y \in X] \\
\theta_{\blacktriangleleft^+} &= x \blacktriangleleft^\star y \vee x \not\approx y \\
\theta_\lhd &= \text{ another tree-walking automaton} \\
\theta_{\text{labels}} &= \text{ as in the old structure } R
\end{aligned}
$$

Top-Down Tree Transducer

A top-down tree transducer (TDTT) is a a tuple $T = \langle Q, \Sigma, \Omega, q_0, P\rangle$ with states Q, ranked alphabets Σ and Ω (input and output), initial state q_0 and a finite set of productions P of the form

$$q(\sigma(x_1,\dots,x_n)) \longrightarrow t$$

where $n \geq 0$, $\sigma \in \Sigma_n$ and $t \in T(\Omega \cup \Sigma(X) \cup Q)$.

The transition relation ($\overset{T}{\Longrightarrow}$) is defined as usual. The transduction realized by a top-down tree transducer T is then defined to be $\{(t_1,t_2) \in T(\Sigma) \times T(\Omega) \mid q_0(t_1) \overset{T}{\Longrightarrow}^* t_2\}$.

$$q_0(a(x_1,x_2)) \longrightarrow b(q_0(x_1), c, q_0(x_2))$$
$$q_0(p) \longrightarrow q$$

Complex Right-Hand-Sides

Let Σ and Ω be ranked alphabets and $n,m \geq 0$. The set of right hand sides $RHS(\Sigma,\Omega,n,m)$ over Σ and Ω with n variables and m parameters is the smallest set $rhs \subseteq T(\Sigma \cup \Omega, X_n \cup Y_m)$ such that

1. $Y_m \subseteq rhs$
2. For $\omega \in \Omega_k$ with $k \geq 0$ and $\varphi_1,\dots,\varphi_k \in rhs$, $\omega(\varphi_1,\dots,\varphi_k) \in rhs$
3. For $q \in Q_{k+1}$ with $k \geq 0$, $x_i \in X_n$ and $\varphi_1,\dots,\varphi_k \in rhs$, $q(x_i,\varphi_1,\dots,\varphi_k) \in rhs$

Macro Tree Transducer

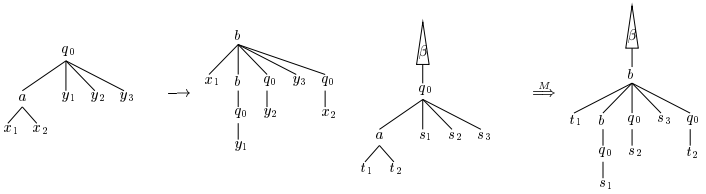
A macro tree transducer (MTT) is a five-tuple $M = \langle Q, \Sigma, \Omega, q_0, P \rangle$ with Q a ranked alphabet of states, ranked alphabets Σ and Ω (input and output), initial state q_0 of rank 1, and a finite set of productions P of the form

$$q(\sigma(x_1,\ldots,x_n), y_1,\ldots,y_m) \longrightarrow t$$

where $n,m \geq 0$, $q \in Q_{m+1}$, $\sigma \in \Sigma_n$ and $t \in RHS(\Sigma,\Omega,n,m)$.

The productions $p \in P$ of M are used as term rewriting rules in the usual way. The transition relation of M is denoted by $\overset{M}{\Longrightarrow}$. The transduction realized by M is the function $\{(t_1,t_2) \in T(\Sigma) \times T(\Omega) \mid (q_0,t_1) \overset{M}{\Longrightarrow}^{*} t_2\}$.

$$q_0(a(x_1,x_2),y_1,y_2,y_3) \longrightarrow b(x_1,b(q_0(y_1)),q_0(y_2),y_3,q_0(x_2))$$

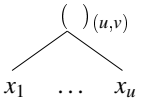


Constructing an MTT

$\sigma \rightsquigarrow \sigma$

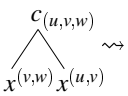
$\pi_i \rightsquigarrow y_i$

$\bullet \rightsquigarrow$

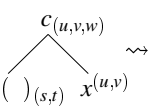


$$\rightsquigarrow q_v^i(\underbrace{(\ )_{(u,v)}(x_1,\ldots,x_u), y_1,\ldots,y_v}_{v+1}) \longrightarrow q_v(\underbrace{x_i, y_1,\ldots,y_v}_{v+1})$$

Constructing an MTT (cont'd)



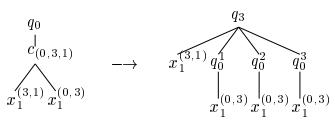
$$q_u(\underbrace{c_{(u,v,w)}(x^{(v,w)},x^{(u,v)}), y_1,\ldots,y_u}_{u+1}) \longrightarrow$$
$$q_v(\underbrace{x^{(v,w)}, q_u^1(x^{(u,v)},y_1,\ldots,y_u),\ldots,q_u^{u-1}(x^{(u,v)},y_1,\ldots,y_u)}_{v+1})$$



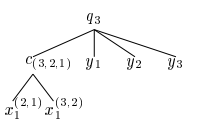
$1 \leq i \leq t$

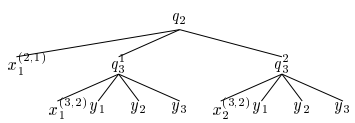
$$q_u^i(\underbrace{c_{(u,v,w)}(x^{(v,w)},x^{(u,v)}), y_1,\ldots,y_u}_{u+1}) \longrightarrow$$
$$q_v^i(\underbrace{x^{(v,w)}, q_u^1(x^{(u,v)},y_1,\ldots,y_u),\ldots,q_u^{u-1}(x^{(u,v)},y_1,\ldots,y_u)}_{v+1})$$

The Example MTT for $\mathcal{G}'$

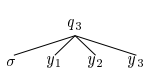


$\longrightarrow \sigma$ for $\sigma \in \{a_{1(0,1)}, a_{2(0,1)}, a_{3(0,1)},\}$



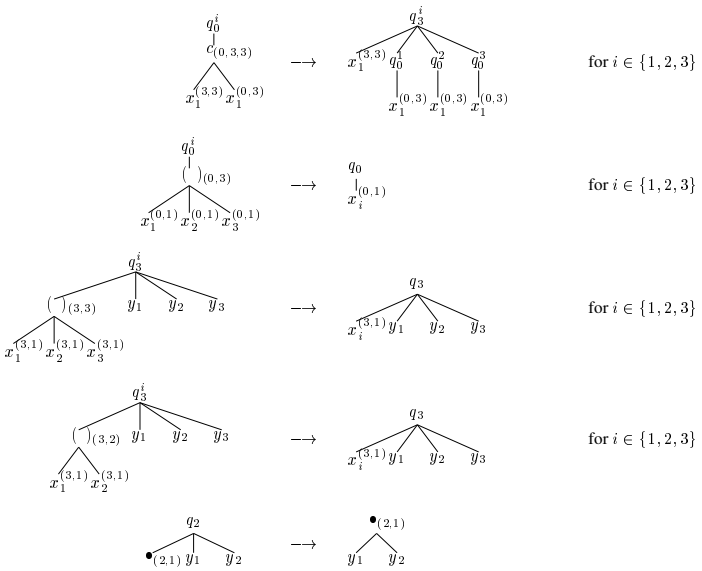


$\longrightarrow y_i$ for $P_i = \pi_i, i \in \{1,2,3\}$



$\longrightarrow \sigma$ for $\sigma \in \{a_{1(3,1)}, a_{2(3,1)}, a_{3(3,1)},\}$

The Example MTT for $\mathcal{G}'$ (cont'd)



MCFGs as a special type of Attribute Grammars

- Every nonterminal has just a single synthesized attribute.
- Attribute evaluation proceeds via a depth-first left-to-right tree traversal.
- The sorted semantic domain consists of the set of words and of tuples of words over some terminal alphabet.
- The semantic operations are simple functions on the domain, i.e., when expressed as explicit terms over the "lifted" signature to be introduced below, each input variable and parameter variable occurs exactly once.

Attribute Grammar

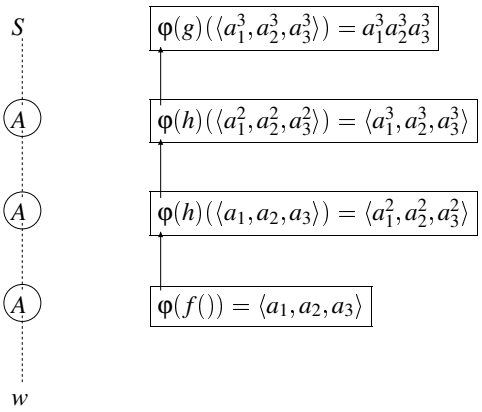
$$G = \langle G_0, D, A, W, R, \alpha_m \rangle$$

G_0	context-free grammar
D	$= \langle \mathcal{S}, \Omega, \Phi, \varphi \rangle$
$\mathcal{S}$	sorts
Ω	sorted domain
Φ	many-sorted signature of operation symbols
φ	interpretation function of Φ
A	set of (inherited/synthesized) attributes
W:	$A \longrightarrow S$ domain assignment
R	set of semantic rules
α_m	the meaning attribute

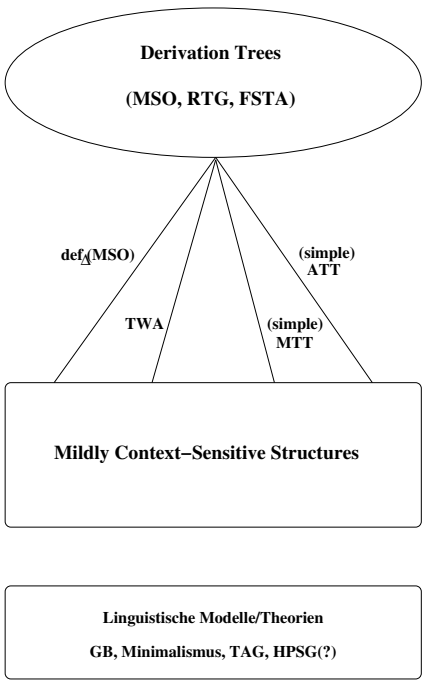
An Example AG

$$\begin{aligned}
G_0 &= \langle \{S,A\}, \{w\}, S, P \rangle \\
P &= \{S \to A,\ A \to w,\ A \to A\} \\
\mathcal{S} &= \{1,3\} \qquad \text{(sorts)} \\
\Omega &= \{\{a_1,a_2,a_3\}^*, (\{a_1,a_2,a_3\}^*)^3\} \\
\Phi &= \{f()_{(0,3)}, g_{(3,1)}, h_{(3,3)}\} \\
\varphi(g)(\langle x,y,z \rangle) &= xyz \\
\varphi(f()) &= \langle a_1,a_2,a_3 \rangle \\
\varphi(h)(\langle x,y,z \rangle) &= \langle xa_1, ya_2, za_3 \rangle \\
A_{syn} &= \{\alpha_s, \alpha_A\} \\
W &= \{\alpha_m, \alpha_s \mapsto 1,\ \alpha_A \mapsto 3\} \\
R &= R(p_1) \cup R(p_2) \cup R(p_3) \\
R(p_1) &= \{\varphi(g)\} \\
R(p_2) &= \{\varphi(f())\} \\
R(p_3) &= \{\varphi(h)\}
\end{aligned}$$

An Example Derivation



Outlook



An Example Grammar $\mathcal{G}_{ww}$: (Non)terminals

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

$$\begin{aligned}
U &= \langle \texttt{b-l}_2, \texttt{-l}_1, - \rangle && \rightsquigarrow \{\omega_i \mid i \in \{1,2\}\} \\
V_i &= \langle \texttt{+L}_1\texttt{c}_i\texttt{-l}_1, \texttt{-l}_1, \texttt{-l}_2 \rangle && \rightsquigarrow \{\varphi_{iu} \mid u \in \{1,2\}^+, i \in \{1,2\}\} \\
U_i &= \langle \texttt{c}_i\texttt{-l}_1, -, \texttt{-l}_2 \rangle && \rightsquigarrow \{\chi_{iu} \mid u \in \{1,2\}^+, i \in \{1,2\}\}
\end{aligned}$$

$$\begin{aligned}
V &= \langle \texttt{+L}_2\texttt{b-l}_2, \texttt{-l}_1, \texttt{-l}_2 \rangle && \rightsquigarrow \{\psi_{iu} \mid u \in \{1,2\}^+, i \in \{1,2\}\} \\
U &= \langle \texttt{b-l}_2, \texttt{-l}_1, - \rangle && \rightsquigarrow \{\omega_{iu} \mid u \in \{1,2\}^+, i \in \{1,2\}\} \\
W_1 &= \langle \texttt{+L}_1\texttt{d}, \texttt{-l}_1, \texttt{-l}_2 \rangle && \rightsquigarrow \{\eta_u \mid u \in \{1,2\}^+\} \\
X_1 &= \langle \texttt{d}, -, \texttt{-l}_2 \rangle && \rightsquigarrow \{\vartheta_u \mid u \in \{1,2\}^+\} \\
W_2 &= \langle \texttt{+L}_2\texttt{c}, -, \texttt{-l}_2 \rangle && \rightsquigarrow \{\kappa_u \mid u \in \{1,2\}^+\} \\
C &= \langle \texttt{c}, -, - \rangle && \rightsquigarrow \{\xi_u \mid u \in \{1,2\}^+\}
\end{aligned}$$

An Example Grammar $\mathcal{G}_{ww}$: Rules

$$i \in \{1,2\}.$$

$$U \to merge_{\langle \beta_i, \alpha_i \rangle}()$$

$$\begin{aligned}
V_i &\to merge_{\langle \gamma_i, \omega_ \rangle}(U) & W_1 &\to merge_{\langle \zeta_1, \omega_ \rangle}(U) \\
U_i &\to move_{\langle \varphi_ \rangle}(V_i) & X_1 &\to move_{\langle \eta_ \rangle}(W_1) \\
V &\to merge_{\langle \delta_i, \chi_ \rangle}(U_i) & W_2 &\to merge_{\langle \zeta_2, \vartheta_ \rangle}(X_1) \\
U &\to move_{\langle \psi_ \rangle}(V) & C &\to move_{\langle \kappa_ \rangle}(W_2)
\end{aligned}$$

$$S \to \pi_1^3(C)$$

An Example Grammar $\mathcal{G}_{ww}$: Basic Functions

$$merge_{\langle \beta_i, \alpha_i \rangle} : (\{1,2\}^*)^0 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle\rangle \mapsto \langle i, i, \varepsilon \rangle$$

$$merge_{\langle \gamma_i, \omega_- \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle i, x_2, x_1 \rangle$$

$$move_{\varphi_-} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle x_2 x_1, \varepsilon, x_3 \rangle$$

$$merge_{\langle \delta_i, \chi_- \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle i, x_1, x_3 \rangle$$

$$move_{\psi_-} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle x_3 x_1, x_2, \varepsilon \rangle$$

$$merge_{\langle \zeta_1, \omega_- \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle \varepsilon, x_2, x_1 \rangle$$

$$move_{\eta_-} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle x_2 x_1, \varepsilon, x_3 \rangle$$

$$merge_{\langle \zeta_2, \vartheta_- \rangle} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle x_1, \varepsilon, x_3 \rangle$$

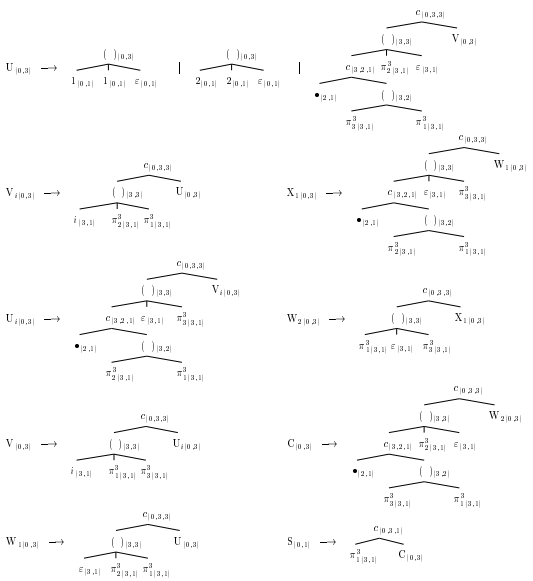
$$move_{\kappa_-} : (\{1,2\}^*)^3 \to (\{1,2\}^*)^3 \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto \langle x_3 x_1, x_2, \varepsilon \rangle$$

$$\pi^3_1 : (\{1,2\}^*)^3 \to \{1,2\}^* \quad \text{with} \quad \langle x_1, x_2, x_3 \rangle \mapsto x_1$$

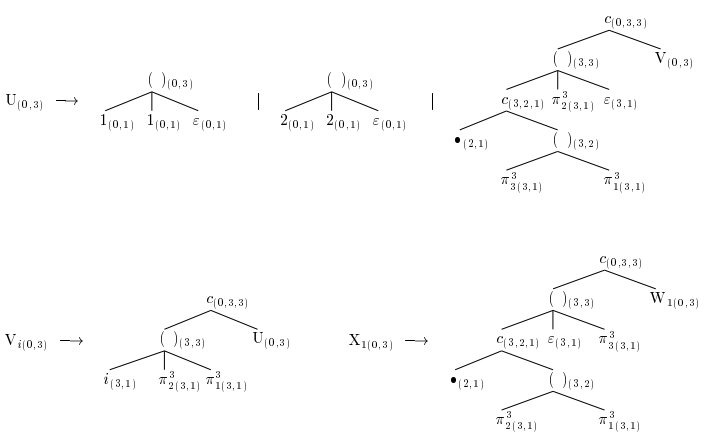
An example derivation

$$\begin{aligned}
&S \\
&\to \pi^3_1(C) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(W_2)) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(X_1))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(move_{\langle \eta_- \rangle}(W_1)))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(move_{\langle \eta_- \rangle}(merge_{\langle \zeta_1, \omega_- \rangle}(U))))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(move_{\langle \eta_- \rangle}(merge_{\langle \zeta_1, \omega_- \rangle}(merge_{\langle \beta_i, \alpha_i \rangle}())))))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(move_{\langle \eta_- \rangle}(merge_{\langle \zeta_1, \omega_- \rangle}(\langle 1,1,\varepsilon \rangle))))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(move_{\langle \eta_- \rangle}(\langle \varepsilon, 1, 1 \rangle))))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(merge_{\langle \zeta_2, \vartheta_- \rangle}(\langle 1\varepsilon, \varepsilon, 1 \rangle))) \\
&\to \pi^3_1(move_{\langle \kappa_- \rangle}(\langle 1\varepsilon, \varepsilon, 1 \rangle)) \\
&\to \pi^3_1(\langle 11\varepsilon, \varepsilon, \varepsilon \rangle) \\
&\to 11\varepsilon
\end{aligned}$$

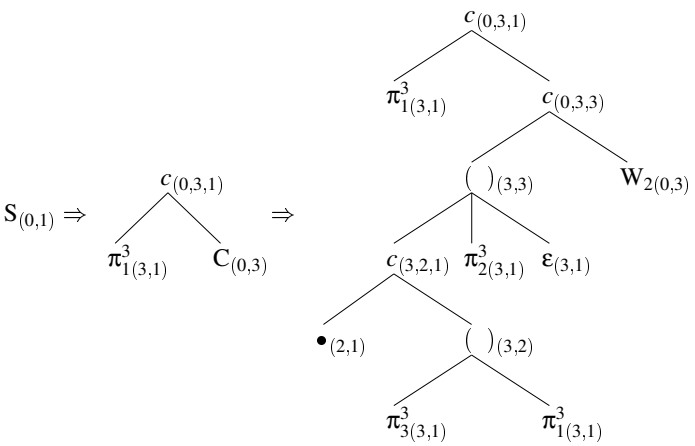
The Translated Example Grammar: $\mathcal{G}'_{ww}$



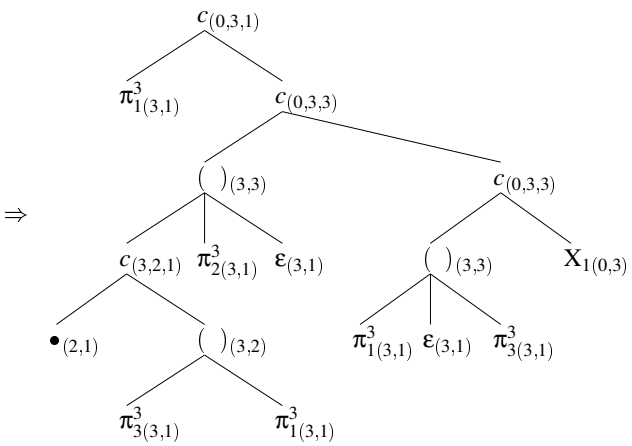
The Translated Example Grammar: $\mathcal{G}'_{ww}$



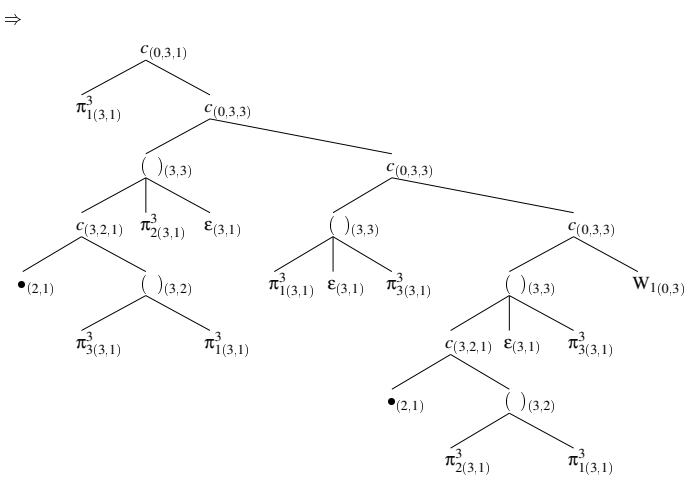
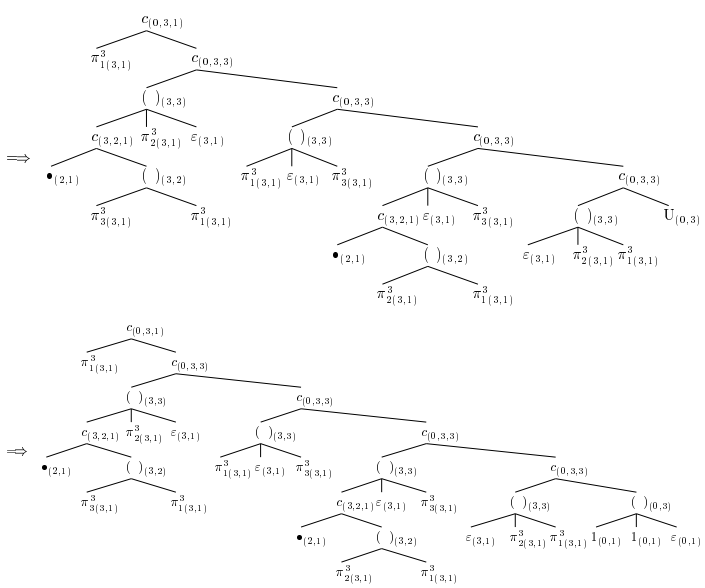
An Example Derivation I



An Example Derivation II



An Example Derivation III



An Example Derivation IV