m-Linear Context-Free Rewriting Systems as Abstract Categorial Grammars

Philippe de Groote, Sylvain Pogodalla

Chapter 8

m-Linear Context-Free Rewriting Systems as Abstract Categorial Grammars

PHILIPPE DE GROOTE* AND SYLVAIN POGODALLA*

ABSTRACT. This paper presents a coding of *m*-linear context-free rewriting systems (*m*-LCFRS) into abstract categorial grammars (ACG). Thus, it shows the latter formalism, which offers a powerful grammatical framework based on a small set of computational primitives, is able to reach some interesting classes of languages w.r.t. natural language modeling.

Introduction

Abstract categorial grammars (ACG) (de Groote 2001) have the property of explicitly generating two languages: an abstract one and an object one. The former may appear as a set of abstract grammatical structures and the latter as the set of the corresponding concrete forms. It then offers a framework in which other grammatical models can be encoded, both in the structures and in the expressions they allow.

This encoding has been done for any G in the class of CFGs (de Groote 2001) or in the classe of TAGs (de Groote 2002). This paper shows such an encoding for *m*-linear context-free rewriting systems (*m*-LCFRS). This enables ACGs to cover important (w.r.t. natural language modeling) classes of languages such as the ones generated by, because of the weak equivalence between them, multicomponent tree adjoining grammars (MCTAGs) (Weir 1988), multiple context-free grammars (MCFG) (Seki et al. 1991) or minimal grammars (MG) (Michaelis 2001).

*INRIA, LORIA, Nancy, France; {Philippe.deGroote,Sylvain.Pogodalla}@loria.fr

Proceedings of Mathematics of Language 8
R. T. Oehrle & J. Rogers (editors).

8.1 Abstract Categorial Grammars

This section defines the notion of an abstract categorial grammar. We first introduce the notions of *linear implicative types*, *higher-order linear signature*, *linear λ-terms* built upon a higher-order linear signature, and *lexicon*.

Definition. *Let A be a set of atomic types. The set $\mathcal{T}(A)$ of* linear implicative types *built upon A is inductively defined as follows:*

1. *if $a \in A$, then $a \in \mathcal{T}(A)$;*
2. *if $\alpha, \beta \in \mathcal{T}(A)$, then $(\alpha \multimap \beta) \in \mathcal{T}(A)$.*

Definition. *A* higher-order linear signature *consists of a triple $\Sigma = \langle A, C, \tau \rangle$, where:*

1. *A is a finite set of atomic types;*
2. *C is a finite set of constants;*
3. *$\tau : C \to \mathcal{T}(A)$ is a function that assigns to each constant in C a linear implicative type in $\mathcal{T}(A)$.*

Definition. *Let X be a infinite countable set of λ-variables. The set $\Lambda(\Sigma)$ of* linear λ-terms *built upon a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$ is inductively defined as follows:*

1. *if $c \in C$, then $c \in \Lambda(\Sigma)$;*
2. *if $x \in X$, then $x \in \Lambda(\Sigma)$;*
3. *if $x \in X$, $t \in \Lambda(\Sigma)$, and x occurs free in t exactly once, then $(\lambda x.t) \in \Lambda(\Sigma)$;*
4. *if $t, u \in \Lambda(\Sigma)$, and the sets of free variables of t and u are disjoint, then $(t\,u) \in \Lambda(\Sigma)$.*

As usual, $\Lambda(\Sigma)$ is provided with notion of capture avoiding α-conversion, substitution and β-reduction (Barendregt 1984).

Given a higher-order linear signature $\Sigma = \langle A, C, \tau \rangle$, each linear λ-term in $\Lambda(\Sigma)$ may be assigned a linear implicative type in $\mathcal{T}(A)$. This type assignment obeys an inference system whose judgements are sequents of the following form:

$$\Gamma \vdash_{\Sigma} t : \alpha$$

where:

1. Γ is a finite set of λ-variable typing declarations of the form '$x : \beta$' (with $x \in X$ and $\beta \in \mathcal{T}(A)$), such that any λ-variable is declared at most once;
2. $t \in \Lambda(\Sigma)$;
3. $\alpha \in \mathcal{T}(A)$.

The axioms and inference rules are the following:

$$\vdash_\Sigma c : \tau(c) \quad \text{(cons)} \qquad \frac{\Gamma, x : \alpha \vdash_\Sigma t : \beta}{\Gamma \vdash_\Sigma (\lambda x.t) : (\alpha \multimap \beta)} \quad \text{(abs)}$$

$$x : \alpha \vdash_\Sigma x : \alpha \quad \text{(var)} \qquad \frac{\Gamma \vdash_\Sigma t : (\alpha \multimap \beta)}{\Gamma, \Delta \vdash_\Sigma (t\,u) : \beta} \quad \text{(app)}$$

Let $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ be two higher-order linear signatures, a lexicon $\mathcal{L} : \Sigma_1 \to \Sigma_2$ is a realization of Σ_1 into Σ_2, i.e., an interpretation of the atomic types of Σ_1 as types built upon A_2 together with an interpretation of the constants of Σ_1 as linear λ-terms built upon Σ_2. These two interpretations must be such that their homomorphic extensions commute with the typing relations. More formally:

Definition. *a* lexicon $\mathcal{L}$ *from* $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ *to* $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ *is defined to be a pair* $\mathcal{L} = \langle F, G \rangle$ *such that:*

1. $F : A_1 \to \mathcal{T}(A_2)$ *is a function that interprets the atomic types of* Σ_1 *as linear implicative types built upon* A_2*;*
2. $G : C_1 \to \Lambda(\Sigma_2)$ *is a function that interprets the constants of* Σ_1 *as linear* λ*-terms built upon* Σ_2*;*
3. *the interpretation functions are compatible with the typing relation, i.e., for any* $c \in C_1$*, the following typing judgement is derivable:*

$$\vdash_{\Sigma_2} G(c) : \hat{F}(\tau_1(c)),$$

where $\hat{F}$ *is the unique homomorphic extension of* F*. Similarly,* $\hat{G}$ *is the unique* λ*-term homomorphism from* $\Lambda(\Sigma_1)$ *to* $\Lambda(\Sigma_2)$ *that extends* G*.*

In the sequel, when '$\mathcal{L}$' will denote a lexicon, it will also denote the homomorphisms $\hat{F}$ and $\hat{G}$ (the intended meaning will be clear from the context).

We are now in a position of defining the notion of abstract categorial grammar.

Definition. *An* abstract categorial grammar *is a quadruple* $\mathcal{G} = \langle \Sigma_1, \Sigma_2, \mathcal{L}, s \rangle$ *where:*

1. Σ_1 *and* Σ_2 *are two higher-order linear signatures; they are called the* abstract vocabulary *and the* object vocabulary*, respectively ;*

2. $\mathcal{L} : \Sigma_1 \to \Sigma_2$ *is a lexicon from the abstract vocabulary to the object vocabulary;*

3. *s is an atomic type of the abstract vocabulary; it is called the* distinguished type *of the grammar.*

The abstract language $\mathscr{A}(\mathcal{G})$ *generated by* $\mathcal{G}$ *is defined as follows:*

$$\mathscr{A}(\mathcal{G}) = \{t \in \Lambda(\Sigma_1) \mid \vdash_{\Sigma_1} t : s \textit{ is derivable}\}$$

In words, the abstract language generated by $\mathcal{G}$ *is the set of closed linear* λ*-terms, built upon the abstract vocabulary* Σ_1*, whose type is the distinguished type s. On the other hand, the* object language $\mathscr{O}(\mathcal{G})$ *generated by* $\mathcal{G}$ *is defined to be the image of the abstract language by the term homomorphism induced by the lexicon* $\mathcal{L}$*:*

$$\mathscr{O}(\mathcal{G}) = \{t \in \Lambda(\Sigma_2) \mid \exists u \in \mathscr{A}(\mathcal{G}).\, t = \mathcal{L}(u)\}$$

8.2 *m*-Linear Context-Free Rewriting Systems

In this section, we directly define the class of *m-linear context-free rewriting systems* (Vijay-Shanker et al. 1987; Weir 1988), even if it can be defined as a proper subclass of the class of *multiple context-free grammars* (Seki et al. 1991; Michaelis 2001), the latter themselves being a subclass of the *generalized context-free grammars* introduced by Pollard (1984).

Definition. *A five-tuple* $G = \langle N, O, F, R, S\rangle$ *is a* m-linear context-free rewriting system (m-LCFRS) *if:*

1. *N is a finite non-empty set of nonterminal symbols;*

2. $O = \bigcup_{i=1}^{m} \langle \Sigma^* \rangle^i$ *for some finite non-empty set* Σ *of terminal symbols with* $\Sigma \cap N = \emptyset$*. O is thet set of all non-empty finite tuples of finite strings in* Σ *such that each tuple has at most m components;*

3. *F is a finite subset of* $\bigcup_{n \leq m} F_n \setminus \{\emptyset\}$ *where* F_n *is the set of partial functions from* $\langle O \rangle^n$ *into O. Moreover, for each* $f \in F$*, there exist* $n(f) \in \mathbb{N}$*,* $d(f) \in \mathbb{N}$ *and* $d_1(f), \ldots, d_{n(f)}(f) \in \mathbb{N}$ *such that:*

$$f : \langle \langle \Sigma^* \rangle^{d_1(f)}, \ldots, \langle \Sigma^* \rangle^{d_{n(f)}(f)} \rangle \mapsto \langle \Sigma^* \rangle^{d(f)} \textit{ and}$$

$$f(\langle\langle x_{11},\dots x_{1d_1(f)}\rangle,\dots,\langle x_{n(f)1},\dots x_{n(f)d_{n(f)}(f)}\rangle\rangle = \langle f_1(\langle y_{11},\dots y_{1n(f_1)}\rangle),\dots, f_{d(f)}(\langle y_{d(f)1},\dots y_{d(f)n(f_{d(f)})}\rangle)\rangle$$

with $\bigcup_{i=1}^{d(f)}\bigcup_{j=1}^{n(f_i)}\{y_{ij}\} = \bigcup_{i=1}^{n(f)}\bigcup_{j=1}^{d_i(f)}\{x_{ij}\}$ *and every* f_i *is linear in each of the* y_{jk}*, i.e.:* $\forall f \in F, \forall i \in [1,d(f)], \exists \xi_{i0},\dots,\xi_{in(f_i)} \in \Sigma^*$ *such that*

$$f_i(\langle y_{i1},\dots y_{in(f_i)}\rangle) = \xi_{i0} y_{i\sigma_i(1)} \xi_{i1} y_{i\sigma_i(2)} \cdots y_{i\sigma_i(n(f_i))} \xi_{in(f_i)}$$

with σ_i *permutation on* $[1,n(f_i)]$*. We call* Ξ *the (finite) set of all the* ξ_{ij} *that are defined in that way.*

4. $R \subset \bigcup_{n\le m}(F\cap F_n)\times N^{n+1}$ *is a finite set of* rewriting rules.
   We usually write a rule $r = \langle f, X_0, X_1,\dots,X_n\rangle \in (F\cap F_n)\times N^{n+1}$ *for some* $n \in \mathbb{N}$ *as* $X_0 \to f(\langle X_1,\dots,X_n\rangle)$*, and* $X_0 \to f()$ *if* $n = 0$*. If* $n = 0$*,* r *is* terminating*, else it is* nonterminating*;*

5. $S \in N$ *is the* distinguished start symbol*;*

6. *there is a function* d_G *from* N *to* $\mathbb{N}$ *such that if* $X_0 \to f(\langle X_1,\dots,X_n\rangle) \in R$*, then* $d(f) = d_G(X_0)$ *and* $d_i(f) = d_G(X_i)$ *where* $d(f)$ *and* $d_i(f)$ *are as in 3;*

7. $d_G(S) = 1$.

We can now define the languages these grammars generate:

Definition. *For each* $X \in N$ *and* $k \in \mathbb{N}$*, the set* $L_G^k(X) \subset O$ *is defined as follows:*

- $L_G^0(X) = \{\theta | X \to f() \in R \text{ and } f() = \theta\}$
- *let* $F_X^n = \{f \in F | \exists X \to f(\langle X_1,\dots,X_n\rangle) \in R\}$*. Then* $L_G^{k+1}(X) = L_G^k(X) \bigcup_{n\le m}\bigcup_{f\in F_X^n} f(\langle L_G^k(X_1),\dots,L_G^k(X_n)\rangle)$

X derives θ in G *if there exist* $X \in N$ *and* $k \in \mathbb{N}$ *such that* $\theta \in L_G^k(X)$*.* θ *is called an* X-phrase in G*. For each* $X \in N$*, the* language derivable from X by G *is* $L_G(X) = \bigcup_{k\in\mathbb{N}} L_G^k(X)$*, and* $L_G(S)$ *is the* language derivable by G.

In addition, we need the definition of the associated parse trees.

Definition. $T = (D_\gamma, V)$ *is a* tree over V *iff* γ *is a function from* D_γ *into* V *where the domain* D_γ *is a finite subset of* $\mathbb{N}^\star$ *such that:*

1. *if* $q \in D_\gamma, p < q,$ *then* $p \in D_\gamma$*;*

2. *if* $p\cdot j \in D_\gamma, j \in \mathbb{N},$ *then* $p\cdot 1, p\cdot 2,\dots,p\cdot(j-1) \in D_\gamma$

where $\mathbb{N}^\star$ *is the free monoid generated by* $\mathbb{N}$, $\cdot$ *is the binary operation,* 0 *is the identity and for* $q \in \mathbb{N}^\star$, $p \leq q$ *iff there is a* $r \in \mathbb{N}^\star$ *such that* $q = p \cdot r$, *and* $p < q$ *iff* $p \leq q$ *and* $p \neq q$.

We say that $D_\gamma = Dom(T)$ *and* $\gamma = Label(T)$.

Definition. *For each* $X \in N$ *and* $k \in \mathbb{N}$, *the set* $PT_G^k(X)$ *of the* parse trees derived from X *is defined as follows:*

- $PT_G^0(X) = \{(\{0\}, \{0 \mapsto (X, f)\}) | X \to f() \in R\}$
- *let* $r_X^n = \{\langle f, X, X_1, \ldots, X_n\rangle \in R\}$. *Then* $PT_G^{k+1}(X) = \bigcup_{n \leq m} \bigcup_{r \in r_X^n} T_r$ *with* $T_r = \{(\{0\} \bigcup \cup_i \{i \cdot D_i\}, \{0 \mapsto (X, f)\} \bigcup \cup_i \{i \cdot \omega \mapsto \gamma_i(\omega)) | (D_i, \gamma_i) \in PT_G^k(X_i)\}$

X derives T in G *if there exists* $X \in N$ *and* $k \in \mathbb{N}$ *such that* $T \in PT_G^k(X)$. T *is called a* X-parse tree in G. *For each* $X \in N$, *the* parse trees derivable from X by G *is* $PT_G(X) = \bigcup_{k \in \mathbb{N}} PT_G^k(X)$, *and* $PT_G = \bigcup_{X \in N} PT_G(X)$ *is the* set of parse trees of G.

Note that from PT_G, we can obviously recover $L_G(X)$ with a linearization function Lin, for all $X \in N$. Indeed, by induction, if $T \in PT_G^0(X)$, it exists $X \to f() \in R$, and with $\mathrm{Lin}(T) = f() = \theta$, $\theta \in L_G^0(X) \subset L_G(X)$. If $T \in PT_G^{k+1}(X)$ there exists $\langle f, X, X_1, \ldots, X_n\rangle \in R$ and $T_i \in PT_G^k(X_i)$. By induction hypothesis, for each $i \leq n$, $\mathrm{Lin}(T_i) \in L_G^k(X_i)$, then $\mathrm{Lin}(T) = f(\langle \mathrm{Lin}(T_1), \ldots, \mathrm{Lin}(T_n)\rangle) = \theta$ is such that $\theta \in L_G^{k+1}(X) \subset L_G(X)$.

8.3 Building an ACG Equivalent to an *m*-LCFRS

In this section, we present the main result of this paper.

Theorem. *For every m-LCFRS* $G = \langle N, O, F, R, S\rangle$, *there exists an ACG* $\mathcal{G}_G$ *such that:*

- *the abstract language* $\mathscr{A}(\mathcal{G}_G)$ *of normal terms is isomorphic to the set of parse-trees of* G;
- *the language generated by* G *coincides with the object language of* $\mathcal{G}_G$, *i.e.* $\mathscr{O}(\mathcal{G}_G) = L_G(S)$.

Proof. First, we add to G a new symbol S' and a new rule $S' \to f_{S'}(S)$ with $f_{S'}(\langle x\rangle) = x$. This is because we model tuples with higher-order functions, and we need to come back to strings at the end. Nevertheless, the generated language is unchanged (we could avoid this by garanteeing that G is not recursive in S).

Then, we define $\mathcal{G}_G = \langle \Sigma_1, \Sigma_2, \mathscr{L}, S'\rangle$ with the abstract vocabulary $\Sigma_1 = \langle A_1, C_1, \tau_1\rangle$ such that:

- $A_1 = N \cup \{S'\}$
- the set of constants C_1 is a set of symbols in one-to-one correspondance with R
- for $c \in C_1$ and $\langle f, X_0, \ldots, X_n \rangle$ the corresponding rule, $\tau_1(c) = X_1 \multimap \cdots \multimap X_n \multimap X_0$

Using the usual encoding of the type σ of strings from an arbitrary atomic type $*$ with $\sigma = * \multimap *$, the empty string being $\lambda x.x$, the string made from one character a being $\lambda x.xa$ and the concatenation operation $+$ being defined as $\lambda f.\lambda g.\lambda x.f(gx)$ (which is an associative operator that admits the identity function as a unit), we can define the object vocabulary as follows (considering σ as an atomic type):

- $A_2 = \{\sigma\}$;
- $C_2 = \{f_1(), \ldots, f_{d(f)}() | \exists X \to f() \in R, f() = \langle f_1(), \ldots, f_{d(f)}() \rangle\} \cup \Xi$;
- τ_2 is defined as assigning the type σ to each $c \in C_2$.

Then we define the lexicon $\mathcal{L}$ with:

- $\mathcal{L}(S') = \sigma$, then for every $X \in N$, $\mathcal{L}(X) = (\underbrace{\sigma \multimap \ldots \multimap \sigma}_{d_G(X) times} \multimap \sigma) \multimap \sigma$ (note that $\mathcal{L}(S) = (\sigma \multimap \sigma) \multimap \sigma$);
- for every $c \in C_1$ that corresponds to a rule $\langle f, X_0, X_1, \ldots, X_n \rangle$ with $X_0 \neq S'$ and

$$f(\langle\langle x_{11}, \ldots x_{1d_1(f)}\rangle, \ldots, \langle x_{n(f)1}, \ldots x_{n(f)d_{n(f)}(f)}\rangle\rangle) = \langle f_1(\langle y_{11}, \ldots y_{1n(f_1)}\rangle), \ldots, f_{d(f)}(\langle y_{d(f)1}, \ldots y_{d(f)n(f_{d(f)})}\rangle)\rangle$$

with $f_i(\langle y_{i1}, \ldots y_{in(f_i)}\rangle) = \xi_{i0} y_{i\sigma_i(1)} \xi_{i1} y_{i\sigma_i(2)} \cdots y_{i\sigma_i(n(f_i))} \xi_{in(f_i)}$

Let $u_i = \xi_{i0} + y_{i\sigma_i(1)} + \xi_{i1} + y_{i\sigma_i(2)} \cdots y_{i\sigma_i(n(f_i))} + \xi_{in(f_i)}$ for each $i \in [1, d(f)]$, then, with $\vec{x}_i = x_{i1} \cdots x_{id_i(f)}$, we have:

$$\mathcal{L}(c) = \lambda T_1 \cdots T_n g.T_1(\lambda \vec{x}_1.T_2(\lambda \vec{x}_2.T_3(\cdots T_n(\lambda \vec{x}_{n(f)}.gu_1 \cdots u_{d(f)}) \cdots)))$$

Indeed, this is a term of $\Lambda(\Sigma_2)$ because of the linearity condition on f and the f_i.

Note that if $c : X$ and X is an atomic type, then f comes from a terminating rule, $f() = \langle f_1(), \ldots, f_{d(f)}() \rangle$ and $\mathcal{L}(c) = \lambda g.g f_1() \cdots f_{d(f)}()$.

If c correspond to the rule $\langle f_{S'}, S', S \rangle$, then $\mathcal{L}(c) = \lambda t.t(\lambda x.x)$.

Then we build I a mapping from the normal terms of $\Lambda(\Sigma_1)$ that are of *atomic types* onto the derivation trees of G by induction as follows:

- if $c \in C_1$, c of type $\alpha \in N$ and correspond to the rule $\alpha \to f()$, $I(c) = (\{0\}, \{0 \mapsto (\alpha, f)\})$
- if $t = cu_1 \cdots u_n$ (in head normal form) is of type $\alpha \in N$ with $c \in C_1$ corresponding to the rule $\langle f, \alpha, \alpha_1, \cdots, \alpha_m \rangle$ where u_i is of type $\alpha_i \in N$, then $m = n$ (because t is of atomic type) and $I(t) = (\{0\} \bigcup_{i=1}^{n} i \cdot \mathrm{Dom}(I(u_i)), \{0 \mapsto (\alpha, f),$ and for all $i \leq n$, $i \cdot w \mapsto \mathrm{Label}(I(u_i))(w))$
- no other case has to be considered since we consider only terms with atomic type.

By induction on the parse trees of G, is it easy to prove that I is an isomorphism. Induction hypothesis $H(n)$: $\forall k \leq n$, $\forall X \in N$, $T \in PT_G^k(X)$ iff there exists a unique $t \in \Lambda(\Sigma_1)$, in normal form and of atomic type, such that $T = I(t)$.

- $n = 0$: $\forall X \in N$, $T \in PT_G^0(X)$ iff there exists $X \to f() \in R$, iff there exists a unique $c \in C_1$ corresponding to $X \to f()$ and c of atomic type X, iff there exists a unique $t = c \in C_1$ such that $T = I(t)$ (by definition of T and $I(t)$);
- $n > 1$: if $k < n$, its trivial by induction hypothesis. If $k = n$, $\forall X \in N$, $T \in PT_G^k(X)$ iff there exists $\langle f, X, X_1, \ldots, X_n \rangle \in R$ and $T_i \in PT_G^{k-1}(X_i)$, iff there exists a unique $c \in C_1$ corresponding to $\langle f, \alpha, \alpha_1, \cdots, \alpha_n \rangle$ and (induction hypothesis) for all $i \leq n$, $\exists! u_i \in \Lambda(\Sigma_1)$ such that $T_i = I(u_i)$, iff there exists a unique $t = cu_1 \cdots u_n$ such that $T = I(t)$ (by definition of T and $I(t)$).

We prove $\mathscr{O}(\mathcal{G}_G) = L_G(S)$ in two steps:

- any tuple $\langle x_1, \ldots, x_n \rangle$ is modeled in $\Lambda(\Sigma_2)$ by $\lambda f. f x_1 \cdots x_n$
- by induction, we prove that for every $t \in \Lambda(\Sigma_1)$ that is of atomic type, $\mathcal{L}(t) = \lambda g. g x_1 \cdots x_n$ where $\langle x_1, \ldots, x_n \rangle = \mathrm{Lin}(I(t))$.

  It is clear if $t = c \in C_1$. If $t = ct_1 \ldots t_n$, with $c \in C_1$ corresponding to the rule $\langle f, X_0, X_1, \cdots, X_n \rangle$, and by induction hypothesis, for each $i \leq n$, $\mathcal{L}(t_i) = \lambda g. g x_{i1} \cdots x_{id_i(f)}$ with $\mathrm{Lin}(I(t_i)) = \langle x_{i1} \cdots x_{id_i(f)} \rangle$.

  So, using the same notations as in the definition of $\mathcal{L}$, we have

$$\begin{aligned}
\mathcal{L}(t) &= \mathcal{L}(c)\mathcal{L}(t_1) \cdots \mathcal{L}(t_n) \\
&= (\lambda T_1 \cdots T_n g. T_1 (\lambda \vec{x}_1. T_2(\cdots)))(\lambda h. h \vec{x_1}) \cdots \mathcal{L}(t_n) \\
&= (\lambda T_2 \cdots T_n g. T_2 (\lambda \vec{x}_2. T_3(\cdots)))\mathcal{L}(t_2) \cdots \mathcal{L}(t_n) \\
&\vdots \\
&= \lambda g. g u_1 \cdots u_{d(f)}
\end{aligned}$$

Since $\mathrm{Lin}(I(t)) = \langle u_1, \ldots, u_n \rangle$ when identifying strings and their coding in $\Lambda(\Sigma_2)$, this ends the proof (by definition of the u_i).

□

8.4 Example

This section provides an example from $G = \langle N, O, F, R, S \rangle$ the 5-LCFRS defined as follows:

- $N = \{A, S\}$
- we have the following rules:

$$\begin{array}{lll} r_0: & S' \to S & f_0(\langle x \rangle) = x \\ r_1: & S \to A & f_1(\langle x_1, \ldots, x_5 \rangle) = \langle x_1 + \cdots + x_5 \rangle \\ r_2: & A \to A & f_2(\langle x_1, \ldots, x_5 \rangle) = \langle x_1 + a, \ldots, x_5 + e \rangle \\ r_3: & A & f_3() = \langle a, b, c, d, e \rangle \end{array}$$

G generates the language $L_G(S') = \{a^n b^n c^n d^n e^n | n > 0\}$.

Following the rules given in the previous section to build the ACG $\mathcal{G}_G$, we have:

$A_1 = \{A, S, S'\}$ $\qquad$ $A_2 = \{\sigma\}$

$C_1 = \{r_0, r_1, r_2, r_3\}$ $\qquad$ $C_2 = \{a, b, c, d, e\}$

τ_1 is such that $\tau_1(r_0) = S \multimap S'$, $\tau_1(r_1) = A \multimap S$, $\tau_1(r_2) = A \multimap A$, $\tau_1(r_3) = A$ $\qquad$ τ_2 is the constant function σ

$$\begin{array}{l} \mathcal{L}(S') = \sigma \\ \mathcal{L}(S) = (\sigma \multimap \sigma) \multimap \sigma \\ \mathcal{L}(A) = (\sigma \multimap \sigma \multimap \sigma \multimap \sigma \multimap \sigma \multimap \sigma) \multimap \sigma \\ \mathcal{L}(r_0) = \lambda t.t(\lambda x.x) \\ \mathcal{L}(r_1) = \lambda T.\lambda g.T(\lambda x_1 x_2 x_3 x_4 x_5.g(x_1 + x_2 + x_3 + x_4 + x_5)) \\ \mathcal{L}(r_2) = \lambda T.\lambda g.T(\lambda x_1 x_2 x_3 x_4 x_5.g(x_1 + a)(x_2 + b)(x_3 + c)(x_4 + d)(x_5 + e)) \\ \mathcal{L}(r_3) = \lambda g.gabcde \end{array}$$

For instance, we can compute:

$$\begin{array}{ll} \mathcal{L}(r_0(r_1(r_2(r_3)))) & = \mathcal{L}(r_0)(\mathcal{L}(r_1)(\mathcal{L}(r_2)(\mathcal{L}(r_3)))) \\ & = \mathcal{L}(r_0)(\mathcal{L}(r_1)(\lambda g.g(a+a)(b+b)(c+c)(d+d)(e+e))) \\ & = \mathcal{L}(r_0)(\lambda g.g(a+a+b+b+c+c+d+d+e+e)) \\ & = a+a+b+b+c+c+d+d+e+e \end{array}$$

Conclusion

This paper gives a coding of *m*-linear context-free rewriting systems into abstract categorial grammars. After the coding of CFGs and TAGs, it shows ACGs to cover still a larger class of languages. Identifying their exact expressive power remains an open problem.

Importantly, it also outlines the ability of ACGs to appear as the kernel of a grammatical framework in which other existing grammatical models may be encoded.

Bibliography

Barendregt, H. P. (1984). *The lambda calculus, its syntax and semantics*. North-Holland. Revised edition.

de Groote, P. (2001). Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pp. 148–155.

de Groote, P. (2002). Tree-adjoining grammars as abstract categorial grammars. In *TAG+6, Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pp. 145–150. Università di Venezia.

Michaelis, J. (2001). Transforming linear context-free rewriting systems into minimalist grammars. In *Proceedings of the conference Logical Aspects of Computational Linguistics (LACL '01)*, volume 2099 of *LNCS/LNAI*.

Pollard, C. (1984). *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*. Ph.D. thesis, Stanford University, CA.

Seki, H., T. Matsumura, M. Fujii, and T. Kasami (1991). On multiple context-free grammars. *Theoretical Computer Science*, **223**:87–120.

Vijay-Shanker, K., D. J. Weir, and A. K. Joshi (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th ACL*, pp. 104–111. Stanford, CA.

Weir, D. J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.