11

Encoding second order string ACG with deterministic tree walking transducers

Sylvain Salvati

Abstract

In this paper we study the class of string languages represented by second order Abstract Categorial Grammar. We prove that this class is the same as the class of output languages of determistic tree walking automata. Together with the result of de Groote and Pogodalla (2004) this shows that the higher-order operations involved in the definition of second order ACGs can always be represented by operations that are at most fourth order.

Keywords Abstract categorial grammar, λ-calculus, deterministic tree walking transducers, mildly context sensitive languages

11.1 Introduction

Abstract Categorial Grammars (ACGs) (de Groote (2001)) are based on the linear logic (Girard (1987)) and on the linear λ-calculus. They describe the surface structures by using for syntax the ideas Montague (1974) devoted to semantics. ACGs describe parse structures with higher-order linear λ-terms and syntax as a higher-order linear homomorphism (lexicon) on parse structures. Intuitively, the higher the order of the parse structures is, the richer should the languages of analysis be and the higher the order of the lexicons is, the richer should the class of languages be. On the one hand, de Groote and Pogodalla (2004) have shown how to encode of several context free formalisms by using second order parse structures (*i.e.* sets of trees). They have encoded Context Free Grammars using second order lexicons, Linear Context Free Tree Grammars using third order lexicons and Linear Context Free Rewriting Systems (Weir (1988)) with fourth order lexicons. On the other

Proceedings of FG-2006.
Editor: Shuly Wintner.

hand Yoshinaka and Kanazawa (2005) have explored the expressivity of lexicalized ACGs. They have exhibited a non-semilinear string language with third order parse structures and an NP-complete string language with fourth order parse structures. (Salvati (2005) gave an example of an NP-complete language with third order parse structures and a first order lexicon).

The present work addresses the problem of the expressivity of ACGs in a particular case. We show that the class of languages defined by second order string ACGs is the same as the class of languages defined as outputs of Deterministic Tree Walking Transducers (DTWT) (Aho and Ullman (1971)). Together with the results of de Groote and Pogodalla (2004) and Weir (1992), this result proves that the generative power of second order string ACGs is exactly the same as the generative power of Linear Context Free Rewriting Systems. This furthermore shows that second order string ACGs can always be described with fourth order lexicons. We may nevertheless conjecture that the use of lexicons of order greater than four may give more compact grammars.

The paper is organized as follows: we first briefly define the linear λ-calculus and ACGs in section 11.2. In section 11.3, we use the correspondence between proofs of linear logic and linear λ-terms to relate sub-formulae of a type α with sub-terms of terms of type α. Section 11.4 introduces h-reduction, the reduction used by the DTWTs which encode second order string ACGs. Section 11.5 presents the encoding of second order string ACGs with DTWTs. Finally we conclude and outline future work in section 11.6.

11.2 Definitions

Given a finite set of atomic types $\mathcal{A}$, we define, $\mathcal{T}_{\mathcal{A}}$, the set of linear applicative types built on $\mathcal{A}$ with the following grammar:

$$\mathcal{T}_{\mathcal{A}} ::= \mathcal{A} \mid (\mathcal{T}_{\mathcal{A}} \multimap \mathcal{T}_{\mathcal{A}})$$

If $\alpha_1, \ldots, \alpha_n$ are elements of $\mathcal{T}_{\mathcal{A}}$ and $\alpha \in \mathcal{A}$ we will write $(\alpha_1, \ldots, \alpha_n) \multimap \alpha$ the type $(\alpha_1 \multimap (\cdots(\alpha_n \multimap \alpha)\cdots))$. The order of the type α, $\mathrm{ord}(\alpha)$, is 1 if α is atomic (*i.e.* $\alpha \in \mathcal{A}$), and $\mathrm{ord}(\alpha \multimap \beta) = \max(\mathrm{ord}(\alpha) + 1, \mathrm{ord}(\beta))$.

Higher-order signatures are triples $(C, \mathcal{A}, \tau)$ where C is a finite set of constants, $\mathcal{A}$ is a finite set of atomic types and τ is a function from C to $\mathcal{T}_{\mathcal{A}}$. The order of a signature $(C, \mathcal{A}, \tau)$ is $\max\{\mathrm{ord}(\tau(a)) | a \in C\}$. Given a higher-order signature $\Sigma = (C, \mathcal{A}, \tau)$ we will denote $\mathcal{A}$ by $\mathcal{A}_\Sigma$, C by C_Σ, τ by τ_Σ and $\mathcal{T}_{\mathcal{A}}$ by $\mathcal{T}_\Sigma$; if $\tau_\Sigma(a) = (\alpha_1, \ldots, \alpha_n) \multimap \alpha$, then the arity of $a \in C_\Sigma$ is n, it will be noted ρ_a^Σ or ρ_a (when Σ is clear from the context).

A higher-order signature Σ is said to be a *string signature* if $\mathcal{A}_\Sigma = \{*\}$, $\# \in C_\Sigma$, $\tau_\Sigma(\#) = *$ and for all $a \in C_\Sigma \backslash \{\#\}$, $\tau_\Sigma(a) = (* \multimap *)$.

We are now going to define the set of linear λ-terms built on a signature

Σ. We assume that the notions of free variables[1], capture-avoiding substitutions, α-conversion, β-reduction, η-reduction... are familiar to the reader. If necessary, one may consult Barendregt (1984).

Given a higher-order signature Σ and $\alpha \in \mathcal{T}_\Sigma$, we assume that we are given an infinite enumerable set of variables x^α, y^α, $z^\alpha \ldots$, Λ_Σ^α the set of linear λ-terms of type α built on Σ is the smallest set verifying:

1. if $a \in C_\Sigma$ and $\tau_\Sigma(a) = \alpha$ then $a \in \Lambda_\Sigma^\alpha$
2. $x^\alpha \in \Lambda_\Sigma^\alpha$
3. if $t_1 \in \Lambda_\Sigma^{(\beta \multimap \alpha)}$, $t_2 \in \Lambda_\Sigma^\beta$ and $FV(t_1) \cap FV(t_2) = \emptyset$ then $(t_1 t_2) \in \Lambda_\Sigma^\alpha$
4. if $t \in \Lambda_\Sigma^\beta$, $x^\alpha \in FV(t)$ then $\lambda x^\alpha.t \in \Lambda_\Sigma^{(\alpha \multimap \beta)}$

The set Λ_Σ denotes $\bigcup_{\alpha \in \mathcal{T}_\Sigma} \Lambda_\Sigma^\alpha$. Linear λ-terms are *linear* because variables may occur free at most once in them and that whenever $\lambda x^\alpha.t$ is a linear λ-term, x^α has exactly one free occurrence in t. Moreover, whenever $t \in \Lambda_\Sigma^\alpha \cap \Lambda_\Sigma^\beta$ then $\alpha = \beta$, *i.e.* every linear λ-term has a unique type in a given signature Σ.

We may, when it is not relevant, strip the typing annotation from the variables. We will write $\lambda x_1 \ldots x_n.t$ for the term $\lambda x_1. \ldots \lambda x_n.t$ and $t_0 t_1 \ldots t_n$ for $(\ldots(t_0 t_1)\ldots t_n)$. Given a list of indices $S = [i_1, \ldots, i_n]$, we will write $\lambda \overrightarrow{x_S}.t$ the term $\lambda x_{i_1} \ldots x_{i_n}.t$, $t_0 \overrightarrow{t_S}$ the term $t_0 t_{i_1} \ldots t_{i_n}$ and $\overrightarrow{c_S} t$ the term $c_{i_1}(\ldots c_{i_n}(t)\ldots)$ when for all $j \in [1, n]$, c_{i_j} has type $* \multimap *$. In particular, $\lambda \overrightarrow{x_n}.t$, $t_0 \overrightarrow{t_n}$ and $\overrightarrow{c_n} t$ may be used when $S = [1, \ldots, n]$.

Given a string signature Σ, strings will be represented by the closed terms of type $*$. For example, the term $c_1(\ldots(c_n\#)\ldots)$ represents the string $c_1 \ldots c_n$; given w, a string built on C_Σ, $/w/$ will denote the term of Λ_Σ^* which is in normal form and represents w.

To define the sub-terms of $t \in \Lambda_\Sigma$, we follow Huet (1997) and consider them as pairs $(C[], t')$ (where $C[]$ is a context, *i.e.* a term with a hole) such that $t = C[t']$. The set of sub-terms of t is denoted by $\mathcal{S}_t$. In particular, we define $\mathcal{S}_t^\alpha$ to be $\{(C[], v) \in \mathcal{S}_t | v \in \Lambda_\Sigma^\alpha\}$. If x is free in t, we note $C_{t,x}[]$ the context such that $C_{t,x}[x] = t$ and x is not free in $C_{t,x}[]$. Remark that since t is linear $C_{t,x}[]$ is always defined.

We say that a term t is in long from if for all $(C[], t') \in \mathcal{S}_t^{\alpha \multimap \beta}$ either $t' = \lambda x.t''$ or $C[] = C'[[]t'']$. Every term can be put in long form by η-expansion, therefore if t is the long form of t', then $t \overset{*}{\rightarrow}_\eta t'$. When a term is in long form, all its possible arguments are abstracted by a λ-abstraction. For example, the term $x^{* \multimap *}$, which is not in long form, can be applied to an argument of type $*$; in long form, this term becomes $\lambda y^*.x^{* \multimap *} y^*$, the possibility of applying it to a term of type $*$ is syntactically represented by the λ-abstraction. A term is in long normal form (lnf for short) if it is both in β-normal form and in long form. The set lnf_Σ^α (*resp.* $\text{clnf}_\Sigma^\alpha$) represents the set of terms of Λ_Σ^α in lnf (*resp.*

[1] Given a λ-term t, we will write $FV(t)$ to denote the set of its free variables.

the closed terms of $\Lambda^{\alpha}_{\Sigma}$ in lnf). In the sequel of the paper we only deal with terms in long form; thus each time we will write $\lambda\overrightarrow{x_S}.t$, $x\overrightarrow{t_S}$ or $a\overrightarrow{t_S}$, we will implicitly make the assumption that t, $x\overrightarrow{t_S}$ or $a\overrightarrow{t_S}$ has an atomic type.

We define homomorphisms between the higher-order signatures Σ_1 and Σ_2 to be pairs (f, g) such that f is a mapping from $\mathcal{T}_{\Sigma_1}$ to $\mathcal{T}_{\Sigma_2}$, and g is a mapping from Λ_{Σ_1} to Λ_{Σ_2}, and verifying:

1. if $\alpha \in \mathcal{A}_{\Sigma_1}$ then $f(\alpha) \in \mathcal{T}_{\Sigma_2}$, otherwise, $f(\alpha \multimap \beta) = f(\alpha) \multimap f(\beta)$
2. for all $a \in C_{\Sigma_1}$ such that $\tau_{\Sigma_1}(a) = \alpha$, $g(a) \in \mathrm{clnf}^{f(\alpha)}_{\Sigma_2}$
3. $g(x^{\alpha}) = x^{f(\alpha)}$
4. $g(t_1 t_2) = g(t_1)g(t_2)$
5. $g(\lambda x^{\alpha}.t) = \lambda x^{f(\alpha)}.g(t)$

One can easily check that whenever $t \in \Lambda^{\alpha}_{\Sigma_1}$, $g(t) \in \Lambda^{f(\alpha)}_{\Sigma_2}$. In general, given a homomorphism $\mathcal{L} = (f, g)$, we will write indistinctly $\mathcal{L}(\alpha)$ for $f(\alpha)$ and $\mathcal{L}(t)$ for $g(t)$. The *order* of $\mathcal{L}$ is $\max\{\mathrm{ord}(\mathcal{L}(\alpha)) | \alpha \in \mathcal{A}_{\Sigma_1}\}$.

An ACG (de Groote (2001)) is a 4-tuple $(\Sigma_1, \Sigma_2, \mathcal{L}, S)$ such that:

1. Σ_1 is a higher-order signature, *the abstract vocabulary*
2. Σ_2 is a higher-order signature, *the object vocabulary*
3. $\mathcal{L}$ is a homomorphism from Σ_1 to Σ_2, *the lexicon*
4. $S \in \mathcal{A}_{\Sigma_1}$

An *abstract constant* (*resp. object constant*) is an element of C_{Σ_1} (*resp.* C_{Σ_2}), an *abstract type* (*resp. object type*) is an element of $\mathcal{T}_{\Sigma_1}$ (*resp.* $\mathcal{T}_{\Sigma_2}$). Given an abstract constant a, $\mathcal{L}(a)$ is called the *realization* of a.

An ACG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{L}, S)$ defines two languages:

1. *the abstract language:* $\mathcal{A}(\mathcal{G}) = \mathrm{clnf}^{S}_{\Sigma_1}$
2. *the object language:* $\mathcal{O}(\mathcal{G}) = \{v \in \mathrm{clnf}_{\Sigma_2} | \exists t \in \mathcal{A}(\mathcal{G}).v =_{\beta\eta} \mathcal{L}(t)\}$

An ACG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{L}, S)$ is said to be a *string ACG* if Σ_2 is a string signature and $\mathcal{L}(S) = *$. The *order of an ACG* is the order of its abstract signature.

11.3 Path in types, active subs-terms and active variables

We assume that we are given a signature Σ and that all the types and all the terms used in this section are built on that signature.

A linear λ-term $t \in \mathrm{lnf}^{\alpha}_{\Sigma}$ represents, via the Curry-Howard isomorphism, a cut-free proof of α in the *Intuitionistic Implicative and Exponential Linear Logic*. This correspondence leads to a natural relation between sub-formulae of α and sub-terms of t. This section presents this relation which will play a central role in our encoding.

The sub-formulae of a type will be designated by means of paths. A path $\pi = i_1 \cdot i_2 \cdots i_{n-1} \cdot i_n$ is a possibly empty sequence of strictly positive integers;

n is the length of π and when $n = 0$, π will be denoted by $\bullet$. Given a set of paths P, $i \cdot P$ denotes the set $\{i \cdot \pi | \pi \in P\}$. The set of paths in the type α, $\mathcal{P}_\alpha$ is defined as follows:

$$\mathcal{P}_{(\alpha_1,\ldots,\alpha_n)\multimap\alpha_0} = \{\bullet\} \cup \bigcup_{i=1}^{n} i \cdot \mathcal{P}_{\alpha_i} \text{ (recall that } \alpha_0 \text{ is atomic)}$$

The set $\mathcal{P}_\alpha$ is split within two parts: the positive paths, denoted by $\mathcal{P}^+_\alpha$ and the negative paths denoted by $\mathcal{P}^-_\alpha$. Positive (*resp.* negative) paths are the path of $\mathcal{P}_\alpha$ which have an even (*resp.* odd) length.

Given a path π, we define $p + \pi$ as: $p + \pi = \begin{cases} \bullet \text{ if } \pi = \bullet \\ (p + k) \cdot \pi' \text{ if } \pi = k \cdot \pi' \end{cases}$

Given $t \in \mathrm{lnf}^\alpha_\Sigma$, we define two particular subsets of $\mathcal{S}_t$, the set of *active subterms*, $\mathcal{AT}_t$, and the set of *active variables*, $\mathcal{AV}_t$. The sets $\mathcal{AT}_t$ and $\mathcal{AV}_t$ are defined as the smallest sets satisfying:

1. $([], t) \in \mathcal{AT}_t$
2. if $(C[], \lambda\overrightarrow{x_n}.t') \in \mathcal{AT}_t$ then for all $i \in [1, n]$,
$$(C[\lambda\overrightarrow{x_n}.C_{t',x_i}[]], x_i) \in \mathcal{AV}_t$$
3. if $(C[[]t_1 \ldots t_n], x) \in \mathcal{AV}_t$ then for all $i \in [1, n]$,
$$(C[xt_1 \ldots t_{i-1}[] \ldots t_n], t_i) \in \mathcal{AT}_t$$

If a term t can be applied to n arguments, then, given $t_1, \ldots, t_n$ terms in lnf, during the β-reduction of $tt_1 \ldots t_n$ the active variables of t will eventually substituted by a term during β-reduction and the residuals of the active subterms of t will eventually become the argument of a redex. On the other hand, the variables of t which are not active will never be substituted and the subterms of t which are not active will never be the argument of a redex.

We can now define two mutually recursive functions $\mathbf{AT}_t$ and $\mathbf{AV}_t$ respectively from $\mathcal{AT}_t$ onto $\mathcal{P}^+_\alpha$ and from $\mathcal{AV}_t$ onto $\mathcal{P}^-_\alpha$:

1. $\mathbf{AT}_t([], t) = \bullet$
2. if $\mathbf{AT}_t(C[], \lambda\overrightarrow{x_n}.t') = \pi$ then for all $i \in [1, n]$,
$$\mathbf{AV}_t(C[\lambda\overrightarrow{x_n}.C_{t',x_i}[]], x_i) = \pi \cdot i$$
3. if $\mathbf{AV}_t(C[[]t_1 \ldots t_n], x) = \pi$ then for all $i \in [1, n]$,
$$\mathbf{AT}_t(C[xt_1 \ldots t_{i-1}[] \ldots t_n], t_i) = \pi \cdot i$$

One can easily check that $\mathbf{AT}_t(C[], v) = \pi$ (*resp.* $\mathbf{AV}_t(C[], x) = \pi$) implies that the type of v (*resp.* x) is the type designated (in the obvious way) by π in α.

The functions $\mathbf{AT}_t$ and $\mathbf{AV}_t$ are bijections whose converse is $\mathbf{P}_t$:

1. $\mathbf{P}_t(\bullet) = ([], t)$
2. $\mathbf{P}_t(\pi \cdot i) = \begin{cases} (C[\lambda\overrightarrow{x_n}.C_{t',x_i}[], x_i) \text{ if } \mathbf{P}_t(\pi) = (C[], \lambda\overrightarrow{x_n}.t') \\ (C[xt_1 \ldots t_{i-1}[] \ldots t_n, t_i) \text{ if } \mathbf{P}_t(\pi) = (C[[]t_1 \ldots t_n], x) \end{cases}$

For all $(C[], t') \in \mathcal{AT}_t$ (*resp.* $(C[], x) \in \mathcal{AV}_t$) it is straightforward that $\mathbf{P}_t(\mathbf{AT}_t(C[], t')) = (C[], t')$ (*resp.* $\mathbf{P}_t(\mathbf{AV}_t(C[], x)) = (C[], x)$); and that for all $\pi \in \mathcal{P}^+_\alpha$ (*resp.* $\pi \in \mathcal{P}^-_\alpha$), $\mathbf{AT}_t(\mathbf{P}_t(\pi)) = \pi$ (*resp.* $\mathbf{AV}_t(\mathbf{P}_t(\pi)) = \pi$).

11.4 *h*-reduction

The DTWTs which encode second order string ACGs perform the normalization of the realization of abstract terms. They use a particular reduction strategy, *h-reduction*, which is related to *head linear reduction* (Danos and Regnier (2004)).

This reduction strategy is only defined for a particular class of λ-terms. Firstly, these λ-terms have to be built on a string signature Σ; secondly, they have a particular form. To describe this form, we need first define $\mathcal{N}^\alpha_\Sigma \subseteq \Lambda^\alpha_\Sigma$ ($\mathcal{N}_\Sigma = \bigcup_{\alpha\in\mathcal{T}_\Sigma} \mathcal{N}^\alpha_\Sigma$) as:

$$\mathcal{N}^\alpha_\Sigma ::= \mathrm{lnf}^\alpha_\Sigma \mid (\mathcal{N}^{\beta\multimap\alpha}_\Sigma \mathcal{N}^\beta_\Sigma)$$

Then, the set of terms we are interested in are the HT-terms defined by the following grammar:

$$\mathcal{HT} ::= \mathcal{N}^*_\Sigma \mid c\mathcal{HT} \mid (\lambda x^{\alpha_1}_1 \dots x^{\alpha_n}_n.\mathcal{HT})\mathcal{N}^{\alpha_1}_\Sigma \dots \mathcal{N}^{\alpha_n}_\Sigma$$

where $c \in C_\Sigma$. Every HT-term is in Λ^*_Σ and is of the form:

$$(\lambda\overrightarrow{x_{S_1}}.\overrightarrow{c_{T_1}}(\dots(\lambda\overrightarrow{x_{S_n}}.\overrightarrow{c_{T_n}}(x_j\overrightarrow{t_Q}))\overrightarrow{v_{S_n}}\dots))\overrightarrow{v_{S_1}}$$

so that $S_i \cap S_j \neq \emptyset$ implies that $i = j$, v_k (with $k \in \bigcup_{i\in[1,n]} S_i$) and t_q (with $q \in Q$) are elements of $\mathcal{N}_\Sigma$.

Given a HT-term,

$$t = (\lambda\overrightarrow{x_{S_1}}.\overrightarrow{c_{T_1}}(\dots(\lambda\overrightarrow{x_{S_n}}.\overrightarrow{c_{T_n}}(x_j\overrightarrow{t_Q}))\overrightarrow{v_{S_n}}\dots))\overrightarrow{v_{S_1}}$$

we say that t *h*-contracts to t' (noted $t \to_h t'$) if

$$t' = (\lambda\overrightarrow{x_{S'_1}}.\overrightarrow{c_{T_1}}(\dots(\lambda\overrightarrow{x_{S'_n}}.\overrightarrow{c_{T_n}}(v_j\overrightarrow{t_Q}))\overrightarrow{v_{S'_n}}\dots))\overrightarrow{v_{S'_1}}$$

where $S'_k = S_k\backslash\{j\}$. It is a routine to check that $t =_\beta t'$, that t' is also a HT-term and that the normal form of t can be obtained in a finite number of *h*-contractions. The reflexive and transitive closure of $\to_h$, *h-reduction*, will be written $\overset{*}{\to}_h$.

Given $\mathcal{G} = (\Sigma_1, \Sigma_2, S, \mathcal{L})$ a second order string ACG, and $u \in \mathrm{clnf}^S_\Sigma$, we are going to see how *h*-contraction normalizes $\mathcal{L}(u)$. The determinism of $\to_h$ allows one to predict statically (*i.e.* without performing the reduction) which sub-term of $\mathcal{L}(u)$ will be substituted to a given bound variable in $\mathcal{L}(u)$ during *h*-reduction. This prediction is based on the notions of *replaceable variables* and *unsafe terms* introduced by Böhm and Dezani-Ciancaglini (1975). Replaceable variables and unsafe terms of u belong to $\mathcal{S}_{\mathcal{L}(u)}$ and will be respectively denoted by $\mathcal{RV}_u$ and $\mathcal{UT}_u$.

If $(C[], a) \in \mathcal{S}_u$ and $(C'[], x) \in \mathcal{AV}_{\mathcal{L}(a)}$, then $(\mathcal{L}(C)[C'[]], x) \in \mathcal{RV}_u$; $\mathcal{UT}_u$ is the smallest set verifying:

1. if $(C[], a\overrightarrow{v_{\rho_a}}) \in \mathcal{S}_u$ and $C[] \neq []$ then $(\mathcal{L}(C)[], \mathcal{L}(a\overrightarrow{v_{\rho_a}})) \in \mathcal{UT}_u$
2. if $(C[], a) \in \mathcal{S}_u$ and $(C'[], v) \in \mathcal{AT}_{\mathcal{L}(a)}$ then $(\mathcal{L}(C)[C'[]], v) \in \mathcal{UT}_u$

The prediction will be given by ϕ_u, a bijection between $\mathcal{RV}_u$ and $\mathcal{UT}_u$. The definition of ϕ_u relies on few more technical definitions.

Given $(C_a[], a) \in \mathcal{S}_u$ such that $C_a[] = C[[]v_1 \dots v_{\rho_a}]$, then

$$(C[av_1 \dots v_{i-1}[] \dots v_{\rho_a}], v_i)$$

is the i^{th} *argument* of $(C_a[], a)$. Given $(C_a[], a), (C_b[], b) \in \mathcal{S}_u$, we say that $(C_a[], a)$ is the *head of the* i^{th} *argument* of $(C_b[], b)$ if

$$C_b[] = C[[]v_1 \dots v_{i-1}(a\overrightarrow{w_{\rho_a}}) \dots v_{\rho_b}] \text{ and } C_a[] = C[bv_1 \dots v_{i-1}([]\overrightarrow{w_{\rho_a}}) \dots v_{\rho_b}]$$

Given $(C[], x) \in \mathcal{RV}_u$, we now define $\phi_u(C[], x)$. As $(C[], x) \in \mathcal{RV}_u$, we have $(C_a[], a) \in \mathcal{S}_u$ and $C_x[]$ such that $(C_x[], x) \in \mathcal{AV}_{\mathcal{L}(a)}$ and $C[] = \mathcal{L}(C_a)[C_x[]]$. Let $\pi = \mathbf{AV}_{\mathcal{L}(a)}(C_x[], x)$, since $\pi \in \mathcal{P}^-_{\mathcal{L}(\tau_{\Sigma_1}(a))}$, π is of odd length, and $\pi = i.\pi'$. Then we have three cases:

1. if $i \leq \rho_a$ and $\pi' = \bullet$, then $\phi_u(C[], x) = (\mathcal{L}(C')[], \mathcal{L}(t))$ where $(C'[], t)$ is the i^{th} argument of $(C_a[], a)$
2. if $i \leq \rho_a$ and $\pi' \neq \bullet$, then $\phi_u(C[], x) = (\mathcal{L}(C_b)[C'[]], t)$ where $(C_b[], b)$ is the head of the i^{th} argument of $(C_a[], a)$ and $(C'[], t) = \mathbf{P}_{\mathcal{L}(b)}(\rho_b + \pi')$
3. if $i > \rho_a$ then $\phi_u(C[], x) = (\mathcal{L}(C_b)[C'[]], t)$ where $(C_a[], a)$ is the head of the k^{th} argument of $(C_b[], b)$ and $(C'[], t) = \mathbf{P}_{\mathcal{L}(b)}(k \cdot (i - \rho_a) \cdot \pi')$.

Computing $\phi_u(C[], x)$ only requires to know about the immediate surrounding of a. This is the reason why the normalization of $\mathcal{L}(u)$ can be performed by a DTWT. To prove the correctness of the prediction of ϕ_u we need the notion of *strict residual*: given t and t' such that $t \overset{*}{\rightarrow}_h t'$, $(C[], v) \in \mathcal{S}_t$ and $(C'[], v) \in \mathcal{S}_{t'}$, we say that $(C'[], v)$ is *the strict residual* of $(C[], v)$ whenever $C[xy_1 \dots y_n] \overset{*}{\rightarrow}_h C'[xy_1 \dots y_n]$ with $FV(v) = \{y_1, \dots, y_n\}$ and x is a fresh variable.

Given t such that $\mathcal{L}(u) \overset{*}{\rightarrow}_h t$, we say that t is *predicted by* ϕ_u if the two following properties hold:

1. for all $(C[], (\lambda\overrightarrow{x_n}\lambda\overrightarrow{y_q}.v)\overrightarrow{v_n}) \in \mathcal{S}_t$ and $i \in [1, n]$, the fact that
$$(C[(\lambda\overrightarrow{x_n}\lambda\overrightarrow{y_q}.C_{v,x_i}[])\overrightarrow{v_n}], x_i)$$
is the strict residual of $(C_{x_i}[], x_i) \in \mathcal{RV}_{\mathcal{L}(u)}$ implies that
$$(C[(\lambda\overrightarrow{x_n}\lambda\overrightarrow{y_q}.v)v_1 \dots v_{i-1}[] \dots v_n], v_i)$$
is the strict residual of $\phi_u(C_{x_i}[], x_i)$.
2. for all $(C[[]\overrightarrow{v_q}], x) \in \mathcal{S}_t$, $(C[[]\overrightarrow{v_q}], x)$ is the strict residual of some $(C'[[]\overrightarrow{v_q}], x) \in \mathcal{RV}_u$.

We are now going to show that h-reduction preserves the predictions of ϕ_u. This will be achieved by using the following technical lemma:

Lemma 25 *Given* $(C[[]\overrightarrow{v_q}], x) \in \mathcal{RV}_u$ *if we have* $\phi_u(C[[]\overrightarrow{v_q}], x) = (C'[], t')$ *then* $t' = (\lambda\overrightarrow{x_p}\overrightarrow{y_q}.w)\overrightarrow{w_p}$ *and we have*

$$\phi_u(C'[(\lambda\overrightarrow{x_p}\overrightarrow{y_q}.C_{w,y_k}[])\overrightarrow{w_p}], y_k) = (C[xv_1 \dots v_{k-1}[] \dots v_q], v_k)$$

Proof.

This proof only consists in unfolding the definitions. Since $(C[[]\overrightarrow{v_q}], x) \in \mathcal{RV}_u$, we must have $(C_a[], a) \in \mathcal{S}_u$ and $C_x[]$ such that:

1. $C[[]\overrightarrow{v_q}] = \mathcal{L}(C_a)[C_x[[]\overrightarrow{v_q}]]$
2. $(C_x[[]\overrightarrow{v_q}], x) \in \mathcal{AV}_{\mathcal{L}(a)}$
3. $\mathbf{AV}_{\mathcal{L}(a)}(C_x[[]\overrightarrow{v_q}], x) = i \cdot \pi$ for some i and π

There are three different cases depending on i and π.

Case 1: $i \leq \rho_a$ and $\pi = \bullet$: this case is very similar to the following one and is thus left to the reader. It is the only case where p may be different from 0.

Case 2: $i \leq \rho_a$ and $\pi \neq \bullet$: by definition if $(C_b[], b)$ is the head of the i^{th} argument of $(C_a[], a)$, and if $\mathbf{P}_{\mathcal{L}(b)}(\rho_b + \pi) = (C''[], \lambda\overrightarrow{y_q}.w)$ then $C'[] = \mathcal{L}(C_b)[C''[]]$ and $t' = \lambda\overrightarrow{y_q}.w$. Let's now suppose that $\pi = m \cdot \pi'$, then we have that $\mathbf{AV}_{\mathcal{L}(b)}(\lambda\overrightarrow{y_q}.C_{w,y_k}[], y_k) = (\rho_b + \pi) \cdot k = (\rho_b + m) \cdot \pi' \cdot k$. Therefore, as $\rho_b + m > \rho_b$ and as $(C_b[], b)$ is the head of the i^{th} argument of $(C_a[], a)$, we have that $\phi_u((\lambda\overrightarrow{y_q}.C_{w,y_k}[]), y_k) = (\mathcal{L}(C_a)[C_k[]], u_k)$ where

$$(C_k[], u_k) = \mathbf{P}_{\mathcal{L}(a)}(i \cdot (\rho_b + m - \rho_b) \cdot \pi' \cdot k) = \mathbf{P}_{\mathcal{L}(a)}(i \cdot \pi \cdot k)$$

But we have that $\mathbf{AV}_{\mathcal{L}(a)}(C_x[[]\overrightarrow{v_q}], x) = i \cdot \pi$ which implies that

$$(C_k[], u_k) = \mathbf{P}_{\mathcal{L}(a)}(i \cdot \pi \cdot k) = (C_x[xv_1 \dots v_{k-1}[] \dots v_r], v_k).$$

Finally as $C[] = \mathcal{L}(C_a)[C_x[]]$ we get the result.

Case 3: $i > \rho_a$: this case is similar to the previous one. □

Proposition 26 *If* $\mathcal{L}(u) \overset{*}{\to}_h t$, *then* t *is predicted by* ϕ_u.

Proof. This proof is done by induction on the number of h-contraction steps of the reduction. The case where this is zero is a simple application of the definitions. Now let's suppose that $\mathcal{L}(u) \overset{*}{\to}_h t \to_h t'$, then, by induction hypothesis, t is predicted by ϕ_u; furthermore, t is a HT-term, thus

$$t = (\lambda\overrightarrow{x_{S_1}}.\overrightarrow{c_{T_1}}(\dots(\lambda\overrightarrow{x_{S_n}}.\overrightarrow{c}_{T_n}(x_j\overrightarrow{t_Q}))\overrightarrow{v_{S_n}}\dots))\overrightarrow{v_{S_1}}$$

and

$$t' = (\lambda\overrightarrow{x_{S'_1}}.\overrightarrow{c_{T_1}}(\dots(\lambda\overrightarrow{x_{S'_n}}.\overrightarrow{c}_{T_n}(v_j\overrightarrow{t_Q}))\overrightarrow{v_{S'_n}}\dots))\overrightarrow{v_{S'_1}}$$

with $S'_i = S_i \backslash \{j\}$.

Within the two conditions required to obtain that t' is predicted by ϕ_u, only the first one requires more than a straightforward application of the induction hypothesis. There is actually only one subterm of t' which is problematic: $v_j\overrightarrow{t_Q}$. From the induction hypothesis we know that the subterm corresponding to x_j in t is the strict residual of $(C[[]\overrightarrow{t_Q}], x_j) \in \mathcal{RV}_u$ and that the subterm corresponding to v_j in t is the strict residual of $\phi_u(C[[]\overrightarrow{t_Q}], x_j)$. Finally the previous lemma allows us to conclude that $v_j\overrightarrow{t_Q}$ fullfills the first condition. □

11.5 Encoding second order string ACGs with DTWT

We are now going to show how to encode second order string ACGs into DTWT. We do not follow the standard definition of DTWT as given in Aho and Ullman (1971). Indeed, instead of walking on the parse trees of a context free grammar, the transducers we use walk on linear λ-terms built on a second order signature. But, as these sets of λ-terms are isomorphic to regular sets of trees, the string languages output by our transducers are the same as those of usual DTWT. By abuse, we call our transducers DTWT.

A DTWT is defined as a 6-tuple

$$\mathbf{A} = (\Sigma, D, Q, T, \delta, \mathbf{q}_0, \mathbf{q}_f)$$

where Σ is a second order signature; $D \in \mathcal{A}_\Sigma$; Q is a finite set of states; T is a finite set of terminals; δ , the transition function, is a partial function from $C_\Sigma \times (Q \backslash \{\mathbf{q}_f\})$ to $(\{up; stay\} \cup (down \times \mathbb{N}^+)) \times Q \times T^*$ where $\mathbb{N}^+$ denotes the set of strictly positive natural numbers and T^* denotes the monoid freely generated by T; $\mathbf{q}_0 \in Q$ is the initial state; and $\mathbf{q}_f \in Q$ is the final state. A *configuration* of $\mathbf{A}$ is given by $(C[], a, \mathbf{q}, s)$ where $C[a] \in \text{clnf}^D_\Sigma$, $a \in C_\Sigma$, $\mathbf{q} \in Q$ and $s \in T^*$; *initial configurations* are of the form $([]\overrightarrow{v_{\rho_a}}, a, \mathbf{q}_0, \epsilon)$ (ϵ being the empty string) where $a\overrightarrow{v_{\rho_a}} \in \text{clnf}^D_\Sigma$. The automaton $\mathbf{A}$ defines a move relation, $\vdash_\mathbf{A}$ ($\vdash^*_\mathbf{A}$ is the reflexive transitive closure of $\vdash_\mathbf{A}$), between configurations: $(C[], a, \mathbf{q}, s) \vdash_\mathbf{A} (C'[], b, \mathbf{q}', sw)$ if $\delta(a, \mathbf{q}) = (\mathbf{q}', m, w)$ and one of the following holds:

1. $m = up$ and $(C[], a)$ is the head of one of the arguments of $(C'[], b)$
2. $m = stay$ and $(C'[], b) = (C[], a)$
3. $m = (down, i)$ and $(C'[], b)$ is the head of the i^{th} argument of $(C[], a)$

Given $a\overrightarrow{v_{\rho_a}} \in \text{clnf}^D_\Sigma$, $a\overrightarrow{v_{\rho_a}}$ generates s with $\mathbf{A}$ if

$$([]\overrightarrow{v_{\rho_a}}, a, \mathbf{q}_0, \epsilon) \vdash^*_\mathbf{A} (C[], b, \mathbf{q}_f, s).$$

The language of $\mathbf{A}$, $\mathbb{L}_\mathbf{A}$, is $\{s | \exists v \in \text{clnf}^D_\Sigma . v \text{ generates } s\}$.

Given a second order string ACG $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathcal{L}, S)$ we are going to build an automaton $\mathbf{A}_\mathcal{G} = (\Sigma, D, Q, T, \delta, \mathbf{q}_0, \mathbf{q}_f)$ such that $\mathcal{O}(\mathcal{G}) = \{/w/ | w \in \mathbb{L}_{\mathbf{A}_\mathcal{G}}\}$. Let $k_\mathcal{G} = \max\{\rho_a | a \in C_{\Sigma_1}\}$, we then define Σ as:

1. $\mathcal{A}_\Sigma = \mathcal{A}_{\Sigma_1} \times [1, k_\mathcal{G}]$

2. $C_\Sigma = C_{\Sigma_1} \times [1, k_\mathcal{G}]$
3. if $\tau_{\Sigma_1}(a) = (\alpha_1, \ldots, \alpha_n) \multimap \alpha$ then

$$\tau_\Sigma((a,k)) = ((\alpha_1, 1), \ldots, (\alpha_n, n)) \multimap (\alpha, k).$$

Remark that if $v \in \mathrm{clnf}_\Sigma^{(\alpha,k)}$, then for all $(C[], (a,j)) \in \mathcal{S}_v$, $C[] \neq []\overrightarrow{v_{\rho_a}}$ implies that $(C[], (a,j))$ is the head of the j^{th} argument of $(C'[], (b,l)) \in \mathcal{S}_v$. Furthermore, given $v = (a,k)\overrightarrow{v_{\rho_a}} \in \mathrm{clnf}_\Sigma^{(\alpha,k)}$ we note $\widetilde{v}$ the term of $\mathrm{clnf}^\alpha_{\Sigma_1}$ such that $\widetilde{v} = a\overrightarrow{\widetilde{v_{\rho_a}}}$.

Then $D = (S, 1)$, $Q = ([0, k_\mathcal{G}] \times P) \cup \{\mathbf{q}_f\}$ where $P = \bigcup_{\alpha \in C_{\Sigma_1}} \mathcal{P}_{\mathcal{L}(\alpha)}$, $\mathbf{q}_0 = (0, \bullet)$; building δ requires some more definitions.

Given (a,k) and (i,π), the *selection path* of (a,k) and (i,π) is:

$$\pi' = \begin{cases} i \cdot \pi \text{ if } i > 0 \\ \rho_a + \pi \text{ if } i = 0 \end{cases}$$

If the selection path of (a,k) and (i,π) is in $\mathcal{P}^+_{\mathcal{L}(\tau_{\Sigma_1}(a))}$ then we say that (a,k) and (i,π) are *coherent*; δ will be only defined on coherent pairs of (a,k) and (i,π). A configuration $K = (C[], (a,k), (i,\pi), w)$ is said to be *coherent* if (a,k) and (i,π) are coherent.

If (a,k) and (i,π) are coherent and if π' is their selection path, then we define the *focused term* of (a,k) and (i,π) as $\mathbf{P}_{\mathcal{L}(a)}(\pi')$. Furthermore, if $(C[], t)$ is the focused term of (a,k) and (i,π) and if $t = \lambda\overrightarrow{x_p}.\overrightarrow{c_n}(x\overrightarrow{v_q})$, then $(C[\lambda\overrightarrow{x_p}.\overrightarrow{c_n}([]\overrightarrow{v_q})], x))$ is called the *focused variable* of (a,k) and (i,π).

If (a,k) and (i,π) are coherent then $\delta((a,k),(i,\pi)) = (\mathbf{q}, \mathit{move}, w)$ depends on the focused term of (a,k) and (i,π), (noted $(C[], t)$):

1. if $t = \overrightarrow{c_n}\#$ then $\mathbf{q} = \mathbf{q}_f$, $\mathit{move} = \mathit{stay}$ and $w = c_1 \ldots c_n$
2. if $t = \lambda\overrightarrow{x_p}.\overrightarrow{c_n}(x\overrightarrow{v_q})$, $\mathbf{AV}_{\mathcal{L}(a)}(C[\lambda\overrightarrow{x_p}.\overrightarrow{c_n}([]\overrightarrow{v_q})], x) = l \cdot \pi''$ and $l > \rho_a$ then $\mathbf{q} = (k, (l - \rho_a) \cdot \pi'')$, $\mathit{move} = \mathit{up}$ and $w = c_1 \ldots c_n$
3. if $t = \lambda\overrightarrow{x_p}.\overrightarrow{c_n}(x\overrightarrow{v_q})$, $\mathbf{AV}_{\mathcal{L}(a)}(C[\lambda\overrightarrow{x_p}.\overrightarrow{c_n}([]\overrightarrow{v_q})], x) = l \cdot \pi''$ and $l \leq \rho_a$ then $\mathbf{q} = (0, \pi'')$, $\mathit{move} = (\mathit{down}, l)$ and $w = c_1 \ldots c_n$

We now relate the walk of $\mathbf{A}_\mathcal{G}$ on $v \in \mathrm{clnf}_\Sigma^{(S,1)}$ with the h-reduction of $\mathcal{L}(\widetilde{v})$. To establish this relation we need to show that the transducer computes $\phi_{\widetilde{v}}$. Given a coherent configuration $K = (C[], (a,k), (i,\pi), w)$, the *activated term* of K is $(\mathcal{L}(C')[], \mathcal{L}(a\overrightarrow{\widetilde{v_{\rho_a}}}))$ if $(i,\pi) = (0, \bullet)$ and $\widetilde{C}[] = C'[[]\overrightarrow{\widetilde{v_{\rho_a}}}]$, otherwise it is $(\mathcal{L}(\widetilde{C})[C'[]], t)$ if $(C'[], t)$ is the focused term of (a,k) and (i,π); the *activated variable* of K is $(\mathcal{L}(\widetilde{C})[C'[]], x)$ if the focused variable of (a,k) and (i,π) is $(C'[], x)$. We will show that given K_1 and K_2 such that $K_1 \vdash_{\mathbf{A}_\mathcal{G}} K_2$, if $(C[], x)$ is the activated variable of K_1 then $\phi_{\widetilde{v}}(C[], x)$ is the activated term of K_2. This property shows that $\mathbf{A}_\mathcal{G}$ performs the h-reduction of $\mathcal{L}(\widetilde{v})$ and that if $\mathcal{L}(\widetilde{v})$ normalizes to $/w/$ then, walking on v, $\mathbf{A}_\mathcal{G}$ ends in the final state and outputs w.

Lemma 27 *Given* $v = (a,1)\overrightarrow{v_{\rho_a}} \in \mathrm{clnf}_{\Sigma}^{(S,1)}$ *and two coherent configurations* K_1 *and* K_2 *such that* $([]\overrightarrow{v_{\rho_a}}, (a,1), (0,\bullet), \epsilon) \vdash^*_{\mathbf{A}_\mathcal{G}} K_1 \vdash_{\mathbf{A}_\mathcal{G}} K_2$*, if* $(C[], x)$ *is the activated variable of* K_1 *then* $\phi_{\overline{v}}(C[], x)$ *is the activated term of* K_2*.*

Proof. As for the proof of lemma 25, this proof is mainly based on the unfolding of the definitions. We simply compute $\phi_{\overline{v}}(C[], x)$ and the activated term of K_2 and then show that they are the same.

We assume that $K_r = (C_r[], (a_r, k_r), (i_r, \pi_r), w_r)$ with $r \in [1,2]$, that π'_r is the selection path of K_r. If $\mathbf{P}_{\mathcal{L}(a_1)}(\pi'_1) = (C'_1[], \lambda\overrightarrow{x_p}.\overrightarrow{c_n}(x\overrightarrow{v_q}))$, then let $\pi''_1 = \mathbf{AV}_{\mathcal{L}(a_1)}(C'_1[\lambda\overrightarrow{x_p}.\overrightarrow{c_n}([]\overrightarrow{v_q}), x)$; as $\pi''_1 \in \mathcal{P}^-_{\mathcal{L}(\tau_{\Sigma_1}(a_1))}$, we know that $\pi''_1 = i \cdot \pi''$. We then have three cases:

Case 1: if $i \leq \rho_{a_1}$ and $\pi'' = \bullet$, then $\phi_{\overline{v}}(C[], x) = (\mathcal{L}(C')[], \mathcal{L}(t))$ if $(C'[], t)$ is the i^{th} argument of $(C_1[], a_1)$. But in that case, we have that $\delta((a_1,k_1),(i_1,\pi_1)) = ((0,\bullet),(down,i),c_1 \ldots c_n)$; thus (a_2,k_2) is the head of the i^{th} argument of (a_1,k_1) and as $(i_2,\pi_2) = (0,\bullet)$, we obtain, by definition, that the activated term of K_2 is indeed $(\mathcal{L}(C')[], \mathcal{L}(t))$.

Case 2: if $i \leq \rho_{a_1}$ and $\pi'' \neq \bullet$, then $\phi_{\overline{v}}(C[], x) = (\mathcal{L}(C_b)[C'[]], t)$ if $(C_b[], b)$ is the i^{th} argument of $(C_1[], a_1)$ and if $(C'[], t) = \mathbf{P}_{\mathcal{L}(b)}(\rho_b + \pi'')$. In that case, we have $\delta((a_1,k_1),(i_1,\pi_1)) = ((0,\pi''),(down,i),c_1 \ldots c_n)$; therefore, (a_2,k_2) is the head of i^{th} argument of (a_1,k_1) which implies that $(C_2[], a_2) = (C_b[], b)$; finally by definition we have that the activated term of K_2 is $(\mathcal{L}(C_b)[C'[]], t) = \phi_{\overline{v}}(C[], x)$.

Case 3: if $i > \rho_{a_1}$ then $\phi_{\overline{v}}(C[], x) = (\mathcal{L}(C_b)[C'[]], t)$ if $(C_{a_1}[], a_1)$ is the head of the $k_1{}^{\text{th}}$ argument of $(C_b[], b)$ and $(C'[], t) = \mathbf{P}_{\mathcal{L}(b)}(k_1 \cdot (i - \rho_{a_1}) \cdot \pi'')$. In that case, we have $\delta((a_1,k_1),(i,\pi)) = ((k_1,(i-\rho_{a_1})\cdot\pi''), up, c_1 \ldots c_n)$, and the definition leads to the fact that the activated term of K_2 is $(\mathcal{L}(C_b)[C'[]], t) = \phi_{\overline{v}}(C[], x)$.

□

Proposition 28 *Given* $u \in \mathrm{clnf}^S_{\Sigma_1}$*, there is a unique* $v = (a,1)\overrightarrow{v_{\rho_a}} \in \mathrm{clnf}^{(S,1)}_{\Sigma}$ *such that* $\widetilde{v} = u$*, and* $([]\overrightarrow{v_{\rho_a}}, (a,1), (0,\bullet), \epsilon) \vdash^*_{\mathbf{A}_\mathcal{G}} (C[], b, q_f, w)$ *iff* $\mathcal{L}(u) =_{\beta\eta} /w/$*.*

Proof. The existence and the uniqueness of v are obvious from the definition of Σ. To prove the proposition it suffices to study the walk of $\mathbf{A}_\mathcal{G}$ on v and the h-reduction of $\mathcal{L}(u)$ in parallel: assume that $K_1 = ([]\overrightarrow{v_{\rho_a}}, (a,1), (0,\bullet), \epsilon)$, $t_1 = \mathcal{L}(u)$, $K_1 \vdash^k_{\mathbf{A}_\mathcal{G}} K_k$ and $t_1 \overset{k}{\to}_h t_k$ (where $\vdash^k_{\mathbf{A}_\mathcal{G}}$ corresponds to k steps of $\mathbf{A}_\mathcal{G}$ and $\overset{k}{\to}_h$ to k steps of h-reduction). The use of the previous lemma and an induction on k prove that t_k is of the form

$$t_k = (\lambda\overrightarrow{x_{S_1}}.\overrightarrow{c_{T_1}}(\ldots(\lambda\overrightarrow{x_{S_k}}.\overrightarrow{c_{T_k}}(x_j\overrightarrow{t_Q}))\overrightarrow{v_{S_k}}\ldots))\overrightarrow{v_{S_1}}$$

if and only if $K_k = (C_k[], (a^k, l_k), (i_k, \pi_k), w_k)$ so that $w_k = \overrightarrow{c_{T_1}} \ldots \overrightarrow{c_{T_{k-1}}}$, if

$(C'_k[], \lambda\overrightarrow{x_{S_k}}.\overrightarrow{c_{T_k}}(x_j\overrightarrow{t_Q})) \in \mathcal{S}_{t_k}$ (with the obvious $C'_k[]$) is the strict residual of $(C''_k[], \lambda\overrightarrow{x_{S_k}}.\overrightarrow{c_{T_k}}(x_j\overrightarrow{t_Q})) \in \mathcal{S}_{t_1}$ then $(C''_k[], \lambda\overrightarrow{x_{S_k}}.\overrightarrow{c_{T_k}}(x_j\overrightarrow{t_Q}))$ is the activated term of K_k and $(C''_k[\lambda\overrightarrow{x_{S_k}}.\overrightarrow{c_{T_k}}([]\overrightarrow{t_Q})], x_j)$ is the activated variable of K_k. This allows us to conclude that the walk ends in the configuration $(C[], b, q_f, w)$ iff $\mathcal{L}(u) =_{\beta\eta} /w/$. □

This finally shows that $\mathcal{O}(\mathcal{G})$ is indeed equal to $\{/w/|w \in \mathbb{L}_{\mathbf{A}_\mathcal{G}}\}$.

11.6 Conclusions and future work

In this paper, we have proved that the languages defined by second order string ACGs were the same as the output languages of DTWT. From the results of Weir (1992) and de Groote and Pogodalla (2004), we obtain as a corollary that the languages defined by second order string ACGs are exactly the languages defined by LCFRS. Furthermore as, according to de Groote and Pogodalla (2004), LCFRS can be encoded by second order string ACGs with a fourth order lexicons, we obtain that every second order string ACG can be encoded by another one whose lexicon has at most fourth order.

In our next work, we would like to exhibit a direct translation of a second order string ACG into another one with a fourth order lexicon. This would help understanding how relevant the order of the lexicon is. We conjecture that using lexicons of order greater than four may lead to more compact grammars. The problem is to know how compact those grammars can be and if the compaction is important whether it can be used do design large grammars for natural languages.

As the tools we used are general, we think it is possible to prove that any second order ACG can be represented as a second order ACG whose lexicon is at most fourth order. Indeed, the notion of paths and the relations they establish with active sub-terms and active variables do not depend on the problem. The only definition which is dependent of the fact we deal with strings is the definition of h-reduction. We nevertheless think that, provided we define a generalized notion of DTWT which would output linear λ-terms instead of strings, we can show that second order ACGs can be encoded with these generalized DTWTs. It would remain to encode those DTWTs with second order ACGs with a fourth order lexicon to generalize our result. But this last part does not seem too difficult.

The first part seems also feasible since it should be possible to generalize h-reduction. Indeed, instead of having a unique variable on which we could make the substitution, the fact that the constants in the term introduce some branching may lead to have several such variables. This would correspond on the generalized DTWTs to the fact that when it would output a branching constant the transducer should duplicate its head in order to have one head to generate each argument of that constant.

Finally this work may lead to the definition of an abstract machine for second order ACGs. Such a machine would be valuable to study the problem of parsing second order ACGs and give insights on the strategies that can be implemented for those grammars. Furthermore, as such a machine would have a language made of linear λ-terms, it would be a first step towards the definition of an abstract machine whose language is a set of λ-terms. In Montague style semantics, the problem of generation mainly consists in parsing languages of λ-terms. We would then obtain a valuable tool to study the problem of generation in that setting.

References

Aho, A. V. and J. D. Ullman. 1971. Translations on a context free grammar. *Information and Control* 19(5):439–475.

Barendregt, Henk P. 1984. *The Lambda Calculus: Its Syntax and Semantics*, vol. 103. Studies in Logic and the Foundations of Mathematics, North-Holland Amsterdam. revised edition.

Böhm, Corrado and Mariangiola Dezani-Ciancaglini. 1975. Lambda-terms as total or partial functions on normal forms. In C. Böhm, ed., *Lambda-Calculus and Computer Science Theory*, vol. 37 of *Lecture Notes in Computer Science*, pages 96–121. Springer. ISBN 3-540-07416-3.

Danos, Vincent and Laurent Regnier. 2004. How abstract machines implement head linear reduction. Preprint of the Institut de Mathématiques de Luminy.

de Groote, Philippe. 2001. Towards abstract categorial grammars. In A. for Computational Linguistic, ed., *Proceedings 39th Annual Meeting and 10th Conference of the European Chapter*, pages 148–155. Morgan Kaufmann Publishers.

de Groote, Philippe and Sylvain Pogodalla. 2004. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information* 13(4):421–438.

Girard, Jean-Yves. 1987. Linear logic. *Theoretical Computer Science* 50:1–102.

Huet, Gérard. 1997. The zipper. *Journal of Functional Programming* 7(5):549–554.

Montague, Richard. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, CT.

Salvati, Sylvain. 2005. *Problèmes de filtrage et problèmes d'analyse pour les grammaires catégorielles abstraites*. Ph.D. thesis, Institut National Polytechnique de Lorraine.

Weir, David Jeremy. 1988. *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania, Philadephia, PA. Supervisor-Aravind K. Joshi.

Weir, David J. 1992. Linear context-free rewriting systems and deterministic tree-walking transducers. In *ACL*, pages 136–143.

Yoshinaka, Ryo and Makoto Kanazawa. 2005. The complexity and generative capacity of lexicalized abstract categorial grammars. In *LACL*, pages 330–346.