Categorial grammars for computing the correspondence between syntax and semantics

An example of interplay between mathematical logic and computational linguistics

Christian Retoré

Université de Bordeaux, LaBRI-CNRS & INRIA

EALING ENS, Paris, Monday September 22nd, 2008

Warning

This lecture is intended to be nothing more than a lecture:

I assume that the prototypic attendant is a student possibly interested in mathematical logic.

Colleagues will be left wanting for more, sorry.

Objective: provide a rather detailed example of relevant use of mathematical logic in formal or computational linguistics by a study of the simplest deductive system handling both syntax and the convergence with semantics: AB grammars, Lambek grammars.

Introduction: the mathematics of computational linguistics

1. Statistics and probability
    - if plain statistics, linguistically frustrating,
    - needs to combine with symbolic methods
    - only makes use of mathematically closed questions (afaik)
2. Formal Language theory, logic (model theory) (Stabler's lecture)
    - slightly beyond context free languages
    - polynomial parsing
    - learnable from positive examples (often left out)
3. ***Logic, proof theory and (typed) lambda calculus***
    - for semantics as expected
    - but also for syntax (intriguing connections with 2)
    - for the correspondence between syntax and semantics

Guidelines

Can some fine mathematics be relevant to formal linguistics?

Can linguistics raise some fruitful motivations to logic?

If language is a computation of some kind can it be depicted by computational logic?

Principles of categorial grammars (CG)

1. Lexical items are mapped into complex categories (lexicon, a.k.a. dictionnary).

2. Universal rules regulate how categories combine.

3. Rules are deductive rules.

1,2: lexicalised grammars

3: ***logical*** CG (Lambek, Moortgat) $\neq$ combinatory CG (Steedman).

Standard history

Rather issued from the logical, philosophical side:

Aristotle, Husserl, ...

Ajdukiewicz (1931) fractions, type checking for formula wellformedness

Bar-Hillel (1953): AB grammars, directionality for depicting word order

Lambek (1958, happy birthday): rules completed into a logical system

van Benthem, Moortgat (1986): models and modal extensions

Subjective history

I would add:

Montague (1970): although he thought that intermediate structures, CG analyses, should disappear (syntax $\longrightarrow$ CG analyses $\longrightarrow$ truth valued models)

Girard (1987), Abrusci (1991): non commutative linear logic, relation to intuitonistic logic, proof nets

Outcomes of categorial grammars

In this order:

- Relation to semantics (in between syntax and semantics)
- Learning algorithms (Buszkowski, Penn, Kanazawa)

More anecdotic, but interesting: Proof nets and human processing (Johnson, Morrill)

Some beautiful results about categorial grammars

1992 Pentus Lambek grammars are context free

1994 Kanazawa convergence of the learning algorithm for AB categorial grammars

2003 Pentus Lambek calculus is NP complete

2007 Salvati Lambek analyses are an Hyper Egde replacement graph language

Current issues in categorial grammars

Extending CG syntax beyond context-freeness, according to linguistic theory e.g. categorial minimalist grammars. (Lecomte, Retoré, Amblard,..),

Practical development of CG for Natural language Processing. (Moortgat, Moot, Steedman, Hockenmaier,..)

Learning classes of categorial grammars from structured data (Tellier, Foret, Bechet,...) or from corpora (Hockenmaier, Moot, ...)

Current issues in categorial grammars Cont’ed

Formal grammar in a type theoretical frame work with a mapping to semantics: abstract categorial grammars (de Groote, Pogodalla, Salvati,...)

Deductions as trees of formal language theory (Tiede, Retoré, Salvati,..)

Various kinds of non commutative or partially commutative linear logic. (Abrusci, Retoré, Ruet,...)

Lexical semantics in a compositional framework

Move from sentence semantics to (short) discourses, compositional DRT (Muskens, de Groote)

Technical contents

- AB grammars
  - definition, examples
  - relation to CFG
  - learnability
- Lambek calculus
  - (without product) definition, examples, properties
  - relation to Montague semantics
  - (with product) definition, properties
  - equivalence with CFG

Classical categorial grammars: AB grammars

Categories a.k.a. types a.k.a fractions:

$$\mathsf{L} ::= \mathsf{P} \quad | \quad \mathsf{L} \setminus \mathsf{L} \quad | \quad \mathsf{L} \,/\, \mathsf{L}$$

P base categories

S (for sentences)

np (for noun phrases)

n (for nouns),

if you wish pp (for prepositional phrase), vp (for verb phrase) etc.

Lexicon/grammar generated language

$\mathrm{Lex}:\quad m$: word /terminal $\longmapsto \mathrm{Lex}(m)$ finite subset of $\mathcal{T}$.

$w_1 \cdots w_n$, is of type u whenever there exists for each w_i a type t_i in $\mathrm{Lex}(w_i)$ such that $t_1 \cdots t_n \longrightarrow u$ with the following reduction patterns:

$$\forall u, v \in \mathsf{L} \qquad \begin{array}{ll} u(u \setminus v) \longrightarrow v & (\setminus_e) \\ (v \,/\, u)u \longrightarrow v & (/_e) \end{array}$$

The set of sentences or the language generated by the grammar is the set of word sequences of type S.

Lexicalised grammar with structured non-terminals
(coherent with modern linguistic theories)

The universal rules are called residuation laws, or simplifications, or elimination rules modus ponens. What do they do?

- If $y : A \setminus B$ then adding any expression $a : A$ on its left yields an expression $ay : B$.
- Symmetrically, if $z : B \ / \ A$ and $a : A$ then $za : B$;

The derivation tree is simply a binary tree whose leaves are the t_i and whose nodes are labeled by rules $/_e$ and $\setminus_e$.

Reduced form for AB grammars

Proposition 1 (Gaifman) *Every AB grammar is equivalent to an AB grammar containing only types of the form*

$$p \qquad (p \mathbin{/} q) \qquad ((p \mathbin{/} q) \mathbin{/} r)$$

where p, q, r *stand for primitive types.*

PROOF : This theorem is an immediate consequence of propositions 3 and 2 to be proved below using Greibach normal form theorem that is now famous. This enables a simpler proof. ◇

Example: a tiny AB grammar

Word	Type(s)
$cosa$	$(S \,/\, (S \,/\, np))$
$guarda$	$(S \,/\, vp)$
$passare$	$(vp \,/\, np)$
il	$(np \,/\, n)$
$treno$	n

$guarda\ passare\ il\ treno : S$ indeed:

$$\begin{array}{l} (S \,/\, vp) \quad (vp \,/\, np) \quad (np \,/\, n) \quad n \\ \longrightarrow (S \,/\, vp) \quad (vp \,/\, np) \quad np \\ \longrightarrow (S \,/\, vp) \quad vp \\ \longrightarrow S \end{array}$$

The derivation tree for this analysis can be written as:

$$[_{/_e}(S \,/\, vp) \quad [_{/_e}(vp \,/\, np) \quad [_{/_e}(np \,/\, n) \quad n]]]$$

$cosa\ guarda\ passare$ not ok, since no rule can reduce the sequence

$$(S \,/\, (S \,/\, np))(S \,/\, vp)(vp \,/\, np)$$

From CFG to AB grammars

Proposition 2 *Every ε-free Context-Free Grammar in Greibach normal form is strongly equivalent to an AB categorial grammar. Thus every ε-free Context-Free grammar is weakly equivalent to an AB categorial grammar.*

PROOF : Let us consider the following AB grammar:

- Its words are the terminals of the CFG.
- Its primitive types are the non terminals of the CFG.
- $\mathrm{Lex}(a)$, the finite set of types associated with a terminal a contains the formulae $((\cdots((X \,/\, X_n) \,/\, X_{n-1}) \,/\, \cdots) \,/\, X_2) \,/\, X_1$ such that there are non terminals $X, X_1, \ldots, X_n$ such that $X \longrightarrow aX_1 \cdots X_n$ is a production rule.

It is then easily observed that the derivation trees of both grammars are isomorphic. ◇

From AB grammars to CFG

Proposition 3 *Every AB grammar is strongly equivalent to a CFG in Chomsky normal form.*

PROOF : Let G be the CFG defined by:

- Terminals T are the words of the AB grammar.
- Non Terminals NT are all the subtypes of the types appearing in the lexicon of the AB grammar — a type is considered to be a subtype of itself.
- The production rules are of two kinds:
  - $X \longrightarrow a$ whenever $X \in \mathrm{Lex}(a)$
  - $X \longrightarrow (X \,/\, Z)Z$ and $X \longrightarrow Z(Z \setminus X)$ for all $X, Z \in NT$ — beware that from the CFG viewpoint $(Z \setminus X)$ or $(X \,/\, Z)$ is a ***single*** non terminal.

◇

Learnability of rigid AB grammars

Rigid: one type per word. Contains non regular languages, but does not contain all of them. Some context free languages, but not even all regular languages.

Rigid AB grammars are learnable from structures from positive examples in Gold sense.

Buskowski & Penn algorithm 1990

Proof of convergence (in Gold sense) by Kanazawa 1994

A learning function maps finite sets of examples to a grammar which generates the examples in the set. Here examples are parse tree as the one produced by an AB, grammar with rules but no type associated with words.

Given any tree language T generated by an AB grammar and any enumeration of it $t_1, t_2,$ ..., there exists an integer N such that for all $n > N$, the language generated by the AB grammar $\phi(\{t_1, \ldots, t_N, \ldots, t_n\})$ is L.

Here language are parse trees, hence examples are trees as well.

- Assign types to the leaves of the examples $\{t_1, \ldots, t_n\}$
- Unify them (make the several types per word one).
- The resulting grammar if any is $\phi(\{t_1, \ldots, t_n\})$

If the examples are from an AB tree language, ϕ of a finite set of examples always exists and can be easily computed.

ϕ converges in the aforementioned sense.

Learning example

(1) $[_{\backslash e}[_{/e}$ a man $]$ swims $]$

(2) $[_{\backslash e}[_{/e}$ a fish $][_{\backslash e}$ swims fast $]]$

Typing:

(3) $[^{S}_{/e}[^{x_2}_{\backslash e}$ a:$(x_2 \,/\, x_1)$ man:$x_1]$ swims:$(x_2 \backslash S)]$

(4) $[^{S}_{\backslash e}[^{y_2}_{/e}$ a:$(y_2 \,/\, y_3)$ fish:$y_3][^{(y_2 \backslash S)}_{\backslash e}$ swims:y_1 fast:$(y_1 \backslash (y_2 \backslash S))]]$

We end up from the previous steps with several types per word. For instance the examples above yields:

(5)

$word$	$type1$	$type2$
a:	$x_2 \,/\, x_1$	$y_2 \,/\, y_3$
fast:		$y_1 \setminus (y_2 \setminus S)$
man:	x_1	
fish:		y_3
swims:	$x_2 \setminus S$	y_1

Unification:

(6)

$$\begin{aligned}
\sigma_u(x_1) &= z_1 \\
\sigma_u(x_2) &= z_2 \\
\sigma_u(y_1) &= z_2 \setminus S \\
\sigma_u(y_2) &= z_2 \\
\sigma_u(y_3) &= z_1
\end{aligned}$$

which yields the rigid grammar/lexicon:

(7)

a:	$z_2 \,/\, z_1$
fast:	$(z_2 \setminus S) \setminus (z_2 \setminus S)$
man:	z_1
fish:	z_1
swims:	$z_2 \setminus S$

Key idea for the convergence

Define $G \sqsubset G'$ as the reflexive relation bewteen G and G' which holds when there exists a substitution σ such that $\sigma(G) \subset G'$ which dos not identify different types of a given word, but this is always the case when the grammar is rigid — $\sigma(G) \subset G'$ is a reflexive relation bewteen G and G' which holds whenever every assignment $a : T$ in G is in G' as well — in particular when G' is rigid, so is G, and they are equal. $\sqsubset$ is an order between grammars.

If the examples arise from a language, unification never fails, but increase the size of the grammar, and there are finitely many grammars below the one to reach.

As opposed to human learning, the language increases instead of specialising to the target language (except using semantics, variant by Tellier).

Limitations of AB-grammars

$(t \,/\, u)$ and $(u \,/\, v)$ does not yield $(t \,/\, v)$

Cosa guarda passare?.

$$(S \,/\, (S \,/\, np))\ (S \,/\, vp)\ (vp \,/\, np) \xrightarrow{(trans.)} (S \,/\, (S \,/\, np)) \quad (S \,/\, np)$$
$$\longrightarrow S$$

that/whom, should be $(n \setminus n) \,/\, (S \,/\, np)$ hence we would need a verb $np \setminus (S \,/\, np)$, unnatural

Maths question: categories = subsets of a free monoid, could there be completeness? (No, see later)

Product free Lambek grammars and calculus

Grammar

With the same notion of lexicon and of grammar,

$w_1 \cdots w_n$, is of type u whenever there exists for each w_i a type t_i in $\mathrm{Lex}(w_i)$ such that $t_1 \cdots t_n \longrightarrow u$

$t_1 \cdots t_n \longrightarrow u$ will be defined by adding extra rules and will be denoted by $t_1 \cdots t_n \vdash u$

Rules

We have two more rules:

this rule requires at least two free hyp.

A left most free hyp.

$$\dfrac{\begin{array}{c} \ldots [A] \ldots\ldots \\ \vdots \\ B \end{array}}{A \backslash B} \backslash_i \text{ binding } A$$

$$\dfrac{\begin{array}{c} \Delta \\ \vdots \\ A \end{array} \quad \begin{array}{c} \Gamma \\ \vdots \\ A \backslash B \end{array}}{B} \backslash_e$$

this rule requires at least two free hyp.

A right most free hyp.

$$\begin{array}{c} \dots\dots\dots[A]\dots \\ \vdots \\ \dfrac{B}{B\ /\ A}\ /_i \text{ binding } A \end{array}$$

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\\ B\ /\ A\end{array}\quad \begin{array}{c}\Delta\\ \vdots\\ A\end{array}}{B}\ /_e$$

Natural deduction in Gentzen style $\neq$ sequent calculus

$$\frac{\Gamma \vdash A \quad \Delta \vdash A \setminus B}{\Gamma, \Delta \vdash B} \setminus_e \qquad \frac{A, \Gamma \vdash C}{\Gamma \vdash A \setminus C} \setminus_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Delta \vdash B \,/\, A \quad \Gamma \vdash A}{\Delta, \Gamma \vdash B} /_e \qquad \frac{\Gamma, A \vdash C}{\Gamma \vdash C \,/\, A} /_i \quad \Gamma \neq \varepsilon$$

$$\frac{}{A \vdash A}\, axiom$$

An example

Here we take up again our small example of an Italian lexicon:

Word	Type(s)
$cosa$	$(S \,/\, (S \,/\, np))$
$guarda$	$(S \,/\, vp)$
$passare$	$(vp \,/\, np)$
il	$(np \,/\, n)$
$treno$	n

Transitivity of $/$, ***Cosa guarda passare*** with the Lambek calculus (we use Natural Deduction in Gentzen style):

$$\dfrac{(S \,/\, (S \,/\, np)) \vdash (S \,/\, (S \,/\, np)) \qquad \dfrac{\dfrac{(S \,/\, vp) \vdash (S \,/\, vp) \qquad \dfrac{(vp \,/\, np) \vdash (vp \,/\, n}{(vp \,/\, np), n}}{(S \,/\, vp), (vp \,/\, np), np \vdash S}}{(S \,/\, vp), (vp \,/\, np) \vdash S \,/\, np}\,/_i}{(S \,/\, (S \,/\, np)), (S \,/\, vp), (vp \,/\, np) \vdash S}\,/_e$$

Variables $\sim$ ***traces*** in chomskyan terms?

Some changes

Allows to reaarrange brackets in the Lambek calculus:
$(a \setminus b) / c \vdash a \setminus (b / c)$, etc.

Fine with object relatives introduced by ***whom/that*** having the type $(n \setminus n) / (S / np)$ with the unique category $(np \setminus S) / np$ for a transitive verb.

$x \vdash (z / x) \setminus z$ and $x \vdash z / (x \setminus z)$ hold for every categories x and z.

From a semantic viewpoint: an np (an individual) can be viewed as a $(S / np) \setminus S$ or $S / (np \setminus S)$ (a function form one place predicates to truth values) that is the set of all the properties of this individual.

The empty sequence

$A \setminus B$ requires an A object before

if $\vdash A$ then the empty sequence can be provided

a: $np \mathbin{/} n$

rather $(n \mathbin{/} n) \mathbin{/} (n \mathbin{/} n)$ ... rather boring: $(n \mathbin{/} n)$

adj: $n \mathbin{/} n$ (provable)

a rather book ???

$$\dfrac{\text{a} : np \mathbin{/} n \qquad \dfrac{\dfrac{\text{rather} : (n \mathbin{/} n) \mathbin{/} (n \mathbin{/} n) \qquad \dfrac{[n]_{\alpha}}{n \mathbin{/} n}\,{/}_{i}{-\alpha}}{n \mathbin{/} n}\,{/}_{e} \qquad \text{book} : n}{n}\,{/}_{e}}{np}\,{/}_{e}$$

Normalization of natural deduction

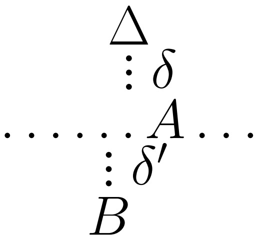
$$\dfrac{\dfrac{\begin{array}{c}\ldots\ldots\ldots[A]_\alpha\ldots\\ \vdots\ \delta'\\ B\end{array}}{B\,/\,A}\,/_i\!-\!\alpha \qquad \begin{array}{c}\Delta\\ \vdots\ \delta\\ A\end{array}}{B}\,/_e \qquad\qquad \dfrac{\begin{array}{c}\Delta\\ \vdots\ \delta\\ A\end{array} \qquad \dfrac{\begin{array}{c}\ldots[A]_\alpha\ldots\ldots\ldots\\ \vdots\ \delta'\\ B\end{array}}{A\,\backslash\,B}\,\backslash_i\!-\!\alpha}{B}\,\backslash_e$$

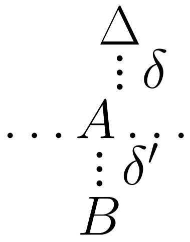
Whenever such a configuration appears, it can be reduced as follows:

1. find the hypothesis A which has been cancelled in the proof δ' of B under some hypotheses including A

2. replace this hypothesis with the proof δ of A

So the configurations above reduce to:





Proposition 4 *Natural deduction for L without product enjoys strong normalization, that is there are no infinite reduction sequences. (size decreases)*

Proposition 5 *Normalization is a locally confluent process. (they do not overlap)*

A ***principal branch*** leading to F a sequence $H_0, \ldots, H_n = F$ of formulae of a natural deduction tree such that:

- H_0 is a free hypothesis
- H_i is the principal premise — the one carrying the eliminated symbol — of an elimination rule whose conclusion is H_{i+1}
- H_n is F

Using this notion, by induction, one obtains:

Proposition 6 *Let d be a normal natural deduction (without product), then:*

1. *if d ends with an elimination then there is a principal branch leading to its conclusion*
2. *each formula in d is the sub-formula of a free hypothesis or of the conclusion*

Here is a proposition of Cohen 1967 needed to prove that every context free grammar is weakly equivalent to a Lambek grammar. It can be easily obtained using the normalisation result, and the subformula property for normal deduction.

Let us call the order $o(A)$ of a formula A the number of alternating implications:

- $o(p) = 0$ when p is a primitive type
- $o(A \setminus B) = \max(o(A) + 1, o(B))$
- $o(B \,/\, A) = \max(o(A) + 1, o(B))$

Proposition 7 *A provable sequent $A_1, \ldots, A_n \vdash p$ of the product free Lambek calculus with $o(A_i) \leq 1$ and p a primitive type is provable with $\setminus_e$ and $/_e$ only — in other words AB derivations and L derivations coincide when types are of order at most one.*

Lambek calculus
and Montague semantics

Semantic types: e: entities, t: truth values

$$types ::= e \quad | \quad t \quad | \quad types{\rightarrow}types$$

Constant	Type
$\exists$	$(e{\rightarrow}t){\rightarrow}t$
$\forall$	$(e{\rightarrow}t){\rightarrow}t$
$\wedge$	$t{\rightarrow}(t{\rightarrow}t)$
$\vee$	$t{\rightarrow}(t{\rightarrow}t)$
$\supset$	$t{\rightarrow}(t{\rightarrow}t)$

and proper constants for the denotation of the words in the lexicon:

<table>
<tr><td>$likes$</td><td>$\lambda x \lambda y\,(\text{likes}\ x)\ y$</td><td>$x : e,\ y : e, \text{likes} : e{\longrightarrow}(e{\longrightarrow}t)$</td></tr>
<tr><td colspan="3">`<< likes >> is a two-place predicate`</td></tr>
<tr><td>$Pierre$</td><td>$\lambda P\,(P\ \text{Pierre})$</td><td>$P : e{\longrightarrow}t,\ \text{Pierre} : e$</td></tr>
<tr><td colspan="3">`<< Pierre >> is viewed as`
`the properties that << Pierre >> holds`</td></tr>
</table>

Higher order logic in simply typed lambda calculus as Church did.

$e{\longrightarrow}t$: e common noun like ***chair*** or an intransitive verb like ***sleep***

$e{\longrightarrow}(e{\longrightarrow}t)$ a transitive verb like ***takes***

Morphism from syntactic types to semantic types:

$(\text{Syntactic type})^*$	$=$	Semantic type	
S^*	$=$	t	a sentence is a proposition
np^*	$=$	e	a noun phrase is an entity
n^*	$=$	$e{\rightarrow}t$	a noun is a subset of the set of entities
$(a \setminus b)^* = (b \,/\, a)^*$	$=$	$a^* \longrightarrow b^*$	extends $(\_)^*$ to all syntactic types

The lexicon associates to each syntactic type $t_k \in \mathrm{Lex}(m)$ of a word m a λ-term τ_k ***whose type is precisely*** t_k^*, the semantic counter part of the syntactic type t_k.

Input for the algorithm computing the semantics

- ***a syntactic analysis of*** $m_1 \ldots m_n$ in Lambek calculus, that is a proof $\mathcal{D}$ of
  $t_1, \ldots, t_m \vdash S$ and

- ***the semantics of each word*** m_1***,. . . et*** m_n, that are λ-terms
  $\tau_i : t_i^*$,

The algorithm computing the semantics

1. Replace every syntactic type in $\mathcal{D}$ with its semantic counterpart; since intuitionistic logic extends the Lambek calculus the result $\mathcal{D}^*$ of this operation is a proof in intuitionistic logic of $t_1^*, \ldots, t_n^* \vdash t = S^*$.
2. Via the Curry-Howard isomorphism, this proof in intuitionistic logic can be viewed as a simply typed λ-term $\mathcal{D}_\lambda^*$ which contains one free variable x_i of type t_i^* per word m_i.
3. Replace in $\mathcal{D}_\lambda^*$. each variable x_i by the λ-term τ_i — whose type is also type t_i^*, so this is a correct substitution.
4. Reduce the resulting λ-term: this provides the semantics of the sentence (another syntactic analysis of the same sentence can lead to a different semantics).

Every normal λ-term of type t without free variables (with solely bound variables and constants) correspond to a formula of predicate calculus.

The previous algorithm relies on;

- embedding of L in intuitionistic logic,
- normalisation of type lambda terms
- predicate logic in typed lambda calculus

Let us see this at work:

word	***syntactic type*** u ***semantic type*** u^* ***semantics :*** λ***-term of type*** u^* x_v ***means that the variable or constant*** x ***is of type*** v
some	$(S \mathbin{/} (np \backslash S)) \mathbin{/} n$ $(e{\longrightarrow}t){\longrightarrow}((e{\longrightarrow}t){\longrightarrow}t)$ $\lambda P_{e\rightarrow t}\ \lambda Q_{e\rightarrow t}\ (\exists_{(e\rightarrow t)\rightarrow t}\ (\lambda x_e(\wedge_{t\rightarrow(t\rightarrow t)}(P\ x)(Q\ x))))$
statements	n $e{\longrightarrow}t$ $\lambda x_e(\texttt{statement}_{e\rightarrow t}\ x)$
speak_about	$(np \backslash S) \mathbin{/} np$ $e{\longrightarrow}(e{\longrightarrow}t)$ $\lambda y_e\ \lambda x_e\ ((\texttt{speak\_about}_{e\rightarrow(e\rightarrow t)}\ x)y)$
themselves	$((np \backslash S) \mathbin{/} np) \backslash (np \backslash S)$ $(e{\longrightarrow}(e{\longrightarrow}t)){\longrightarrow}(e{\longrightarrow}t)$ $\lambda P_{e\rightarrow(e\rightarrow t)}\ \lambda x_e\ ((P\ x)x)$

Let us first show that ***Some statements speak about themselves.*** belongs to the language generated by this lexicon. So let us prove (in natural deduction) the following :

$$(S\,/\,(np\,\backslash\,S))\,/\,n\ ,\ n\ ,\ (np\,\backslash\,S)\,/\,np\ ,\ ((np\,\backslash\,S)\,/\,np)\,\backslash\,(np\,\backslash\,S) \vdash S$$

using the abbreviations So (some) Sta (statements) SpA (speak about) $Refl$ (themselves) for the syntactic types :

$$\dfrac{\dfrac{So \vdash (S/(np\backslash S))/n \qquad Sta \vdash n}{So, Sta \vdash (S/(np\backslash S))}\,/_e \qquad \dfrac{SpA \vdash (np\backslash S)/np \qquad Refl \vdash ((np\backslash S)/np}{SpA, Refl \vdash (np\backslash S)}}{So, Sta, SpA, Refl \vdash S}$$

Using the homomorphism from syntactic types to semantic types we obtain the following intuitionistic deduction, where $So^*, Sta^*, SpA^*, Refl^*$ are abbreviations for the semantic types respectively associated with the syntactic types: $So, Sta, SpA, Refl$:

$$\dfrac{\dfrac{So^* \vdash (e\to t)\to(e\to t)\to t \quad Sta^* \vdash e\to t}{So^*, Sta^* \vdash (e\to t)\to t}\to_e \qquad \dfrac{SpA^* \vdash e\to e\to t \quad Refl^* \vdash (e-}{SpA^*, Refl^* \vdash e-}}{So^*, Sta^*, SpA^*, Refl^* \vdash t}$$

The λ-term representing this deduction simply is

((some statements) (themsleves speak_about)) of type t

where *some,statements,themselves,speak_about* are variables with respective types

$$So^*, Sta^*, Refl^*, SpA^*$$

Let us replace these variables with the semantic λ-terms (of the same type) which are given in the lexicon. We obtain the following λ-term of type t (written on two lines) that we reduce:

$$\Big(\big(\lambda P_{e\to t}\,\lambda Q_{e\to t}\,(\exists_{(e\to t)\to t}\,(\lambda x_e(\wedge(P\,x)(Q\,x)))))\big)\big(\lambda x_e(\texttt{statement}_{e\to t}\,x$$

$$\Big(\big(\lambda P_{e\to(e\to t)}\,\lambda x_e\,((P\,x)x)\big)\big(\lambda y_e\,\lambda x_e\,((\texttt{speak\_about}_{e\to(e\to t)}\,x)y)\big)\Big)$$

$$\downarrow\beta$$

$$\big(\lambda Q_{e\to t}\,(\exists_{(e\to t)\to t}\,(\lambda x_e(\wedge_{t\to(t\to t)}(\texttt{statement}_{e\to t}\,x)(Q\,x)))))\big)$$
$$\big(\lambda x_e\,((\texttt{speak\_about}_{e\to(e\to t)}\,x)x)\big)$$

$$\downarrow\beta$$

$$\big(\exists_{(e\to t)\to t}\,(\lambda x_e(\wedge(\texttt{statement}_{e\to t}\,x)((\texttt{speak\_about}_{e\to(e\to t)}\,x)x))))\big)$$

The previous term represent the following formula of predicate calculus (in a more pleasant format) :

$$\exists x : e \left(\texttt{statement}(x) \land \texttt{speak\_about}(x, x)\right)$$

This is the semantics of the analysed sentence.

Discussion

Fine point: compositionality at work with neat logical tools.

Technical difficulty: how to extend the syntax while keeping this correspondence? Abstract Categorial Grammars (for tree grammars) or Categorial Minimalist Grammars.

IMHO Intermediate language (despised by Montague) much more exciting than models and possible worlds.

What would be a good interpretation?

Discourse and DRT like objections solved by λ DRT or de Groote with contexts as argument.

IMHO main problem is lexical semantics: how to integrate relation between the predicates and constants in the model. It is related to a good interpretation without too much unfolding without too much knowledge representation.

Lambek syntactic calculus 2: with product, sequent calculus

Also at the end we wont use it, the proof of the context-freeness of Lambek languages is easier with product and sequent calculus. SO we have a new connective $A \bullet B$ which means an A followed by a B.

$$\mathsf{Lp} ::= \mathsf{P} \quad | \quad \mathsf{Lp} \setminus \mathsf{Lp} \quad | \quad \mathsf{Lp} \,/\, \mathsf{Lp} \quad | \quad \mathsf{Lp} \bullet \mathsf{Lp}$$

Sequent calculus

$$\frac{\Gamma, B, \Gamma' \vdash C \quad \Delta \vdash A}{\Gamma, \Delta, A \backslash B, \Gamma' \vdash C}\,\backslash_h \qquad\qquad \frac{A, \Gamma \vdash C}{\Gamma \vdash A \backslash C}\,\backslash_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Gamma, B, \Gamma' \vdash C \quad \Delta \vdash A}{\Gamma, B / A, \Delta, \Gamma' \vdash C}\,/_h \qquad\qquad \frac{\Gamma, A \vdash C}{\Gamma \vdash C / A}\,/_i \quad \Gamma \neq \varepsilon$$

$$\frac{\Gamma, A, B, \Gamma' \vdash C}{\Gamma, A \bullet B, \Gamma' \vdash C}\,\bullet_h \qquad\qquad \frac{\Delta \vdash A \quad \Gamma \vdash B}{\Delta, \Gamma \vdash A \bullet B}\,\bullet_i$$

$$\frac{\Gamma \vdash A \quad \Delta_1, A, \Delta_2 \vdash B}{\Delta_1, \Gamma, \Delta_2 \vdash B}\,cut \qquad\qquad \frac{}{A \vdash A}\,axiom$$

Here is an obvious proposition, known as η-expansion:

Proposition 8 *Every axiom $A \vdash A$ can be derived from axioms $p \vdash p$, with p being a primitive type (and the proof does not use the cut rule).*

The polarity of an occurrence of a propositional variable p in a formula is defined as usual:

- p is positive in p
- if p is positive in A, then
  - p is positive in $X \bullet A, \quad A \bullet X, \qquad X \backslash A, \quad A \,/\, X$
  - p is negative in $A \backslash X, \quad X \,/\, A$
- if p is negative in A, then
  - p is negative in $X \bullet A, \quad A \bullet X, \qquad X \backslash A, \quad A \,/\, X$
  - p is positive in $A \backslash X, \quad X \,/\, A$

The polarity of an occurrence of a propositional variable p in a sequent $\Gamma \vdash C$ is:

- if p is in C, the polarity of p in C
- if p is in a formula G of Γ, the opposite of the polarity of p in G.

One can use only axioms $p \vdash p$ with p a primitive type.

Proposition 9 *Each propositional variable has exactly the same number of positive and negative occurrences in a provable sequent.*

An example

Cosa guarda passare : bis repetita placent

$$\dfrac{S \vdash S \qquad \dfrac{\dfrac{\dfrac{S \vdash S \quad vp \vdash vp}{S \,/\, vp, vp \vdash S}\,/_h \qquad np \vdash np}{S \,/\, vp, vp \,/\, np, np \vdash S}\,/_h}{S \,/\, vp, vp \,/\, np \vdash S \,/\, np}\,/_i}{(S \,/\, (S \,/\, np)), S \,/\, vp, vp \,/\, np \vdash S}\,/_h$$

Cut-elimination for sequent calculus

Proposition 10 *In a cut-free proof of* $A_1, \ldots, A_n \vdash A_{n+1}$ *every formula of every sequent is a sub-formula of some formula* A_i *(*$1 \leq i \leq n+1$*).*

PROOF : By case inspection it is easily observed that every rule of the sequent calculus but the cut rule, satisfies the property that every formula in its premise sequent(s) is a sub-formula of some formula in its conclusion sequent. $\diamond$

Proposition 11 *Every proof of a given sequent $\Gamma \vdash A$ can be turned into a cut free proof of the same sequent — all formulae in the cut-free proof being sub-formulae of the sequent $\Gamma \vdash C$.*

PROOF : Much easier than for classical logic: no contraction, no weakening. We can permute rules for having dual rules both creating a cut of maximal degree, and then one of the reduction patterns applies. $\diamond$

Proposition 12 *There is an algorithm which decides whether a sequent is derivable in L. This result can be viewed as a converging parsing algorithm.*

PROOF : Assume we want to prove a sequent. Since the cut rule is not needed, we have finitely many rules to try, each of these rules leading to prove one or two smaller sequents $\diamond$

Tthe interpolation theorem for Lambek calculus which appeared in the thesis of Roorda . It's somehow the converse of cut-elimination, proving by axioms and cuts... as a CFG. It can be proved for product free Lambek calculus, but it's more difficult. Here, a boring induction is enough.

Proposition 13 *Let $\Gamma, \Delta, \Theta \vdash C$ be a provable sequent in L, with $\Delta \neq \varepsilon$. There exists an interpolant of Δ that is a formula I such that:*

1. $\Delta \vdash I$
2. $\Gamma, I, \Theta \vdash C$
3. $\rho_p(I) \leq \rho_p(\Delta)$ *for every primitive type p*
4. $\rho_p(I) \leq \rho_p(\Gamma, \Theta, C)$ *for every primitive type p*

Equivalence of sequent calculus and natural deduction

From natural deduction to sequent calculus

From natural deduction to sequent calculus

(One can do better)

Replace:

with:

$$\frac{\Delta \vdash A \quad \Gamma \vdash A \backslash B}{\Delta, \Gamma \vdash B}\,\backslash_e$$

$$\frac{\Gamma \vdash A \backslash B \qquad \dfrac{\Delta \vdash A \qquad \dfrac{}{B \vdash B}\,ax}{\Delta, A \backslash B \vdash B}\,\backslash_h}{\Delta, \Gamma \vdash B}\,cut$$

$$\frac{\Gamma \vdash B \,/\, A \quad \Delta \vdash A}{\Gamma, \Delta \vdash B}\,/_e$$

$$\frac{\Gamma \vdash B \,/\, A \qquad \dfrac{\Delta \vdash A \qquad \dfrac{}{B \vdash B}\,ax}{B \,/\, A, \Delta \vdash B}\,/_h}{\Gamma, \Delta \vdash B}\,cut$$

$$\frac{\Gamma \vdash A \bullet B \quad \Delta, A, B, \Theta \vdash C}{\Delta, \Gamma, \Theta \vdash C}\,\bullet_e$$

$$\frac{\Gamma \vdash A \bullet B \qquad \dfrac{\Delta, A, B, \Theta \vdash C}{\Delta, A \bullet B, \Theta \vdash C}\,\bullet_h}{\Delta, \Gamma, \Theta \vdash C}\,cut$$

From sequent calculus to natural deduction

From sequent calculus to natural deduction

By induction on the height of a sequent calculus proof, let us see that it can be turned into a natural deduction. As above, we will not exhibit a translation from cut free proofs to normal deductions, although it is possible.

- If the proof consists in an axiom, its translation is obvious.
- If the proof ends with an introduction rule, $\backslash_i$, $/_i$ or $\bullet_i$ by induction hypothesis we have a deduction of the premise(s) and as these rules also exist in natural deduction and the translation is obvious.

- If the proof ends with an $\backslash_h$ rule:

$$\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma, B, \Gamma' \vdash C\end{array} \qquad \begin{array}{c}\vdots\,\delta\\ \Delta \vdash A\end{array}}{\Gamma, \Delta, A \backslash B, \Gamma' \vdash C}\,\backslash_h$$

then by induction hypothesis we have two natural deduction proofs, γ^* of $\Gamma, B, \Gamma' \vdash C$ and δ^* of $\Delta \vdash A$ and a translation of the whole proof is:

$$\begin{array}{ccc} \Gamma & \begin{array}{c} \dfrac{\begin{array}{c}\Delta\\ \vdots\,\delta\\ A\end{array} \quad A \backslash B}{B}\,\backslash_e \\ \vdots\,\gamma \\ C\end{array} & \Gamma' \end{array}$$

- If the proof ends with $/_h$ we proceed symmetrically.

- If the proof ends with $\bullet_h$:

$$\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma, A, B, \Gamma' \vdash C\end{array}}{\Gamma, A \bullet B, \Gamma' \vdash C}\,\bullet_h$$

by induction hypothesis we have a proof γ^* of $\Gamma, A \bullet B, \Gamma' \vdash C$ and a translation is the following:

$$\dfrac{A \bullet B \qquad \begin{array}{c}\Gamma \quad A\ B \quad \Gamma'\\ \vdots\,\gamma^*\\ C\end{array}}{C}\,\bullet_e$$

- If the proof ends with a cut:

$$\dfrac{\begin{array}{c}\vdots\,\gamma\\ \Gamma\vdash X\end{array}\qquad\begin{array}{c}\vdots\,\delta\\ \Delta, X, \Delta'\vdash C\end{array}}{C}\,cut$$

by induction hypothesis we have two natural deductions $\gamma^*$of $\Gamma\vdash X$ and δ^* of $\Delta, X, \Delta'\vdash C$ and a translation is:

$$\begin{array}{ccc} & \Gamma & \\ & \vdots\,\gamma^* & \\ \Delta & X & \Delta' \\ & \vdots\,\delta & \\ & C & \end{array}$$

Models for the Lambek calculus: soundness and completeness

Residuated semi-groups and the free group model

Let us call a ***residuated semi-group***, a structure $(M, \circ, \backslash\backslash, //, \sqsubset)$ where

- M is a set.
- $\circ$ is an associative composition over M — $(M, \circ)$ is a semi-group.
- $\backslash\backslash$ and $//$ are binary composition law on M.
- $\sqsubset$ is an order on M.

Satisfying a the property (RSG) of next slide.

(RSG) The following order relations are either all true or all false:

$$\begin{aligned} a \quad &\sqsubseteq (c \mathbin{/\!/} b) \\ (a \circ b) &\sqsubseteq c \\ b &\sqsubseteq (a \mathbin{\backslash\!\backslash} c) \end{aligned}$$

Proposition 14 *In a residuated semi-group* $(M, \circ, \backslash\!\backslash, /\!/, \sqsubset)$*, for all* $a, b, x, y \in M$ *one has:*

1. $a \sqsubset b \quad \Rightarrow \quad (a \circ x) \sqsubset (b \circ x)$
2. $a \sqsubset b \quad \Rightarrow \quad (x \circ a) \sqsubset (x \circ b)$
3. $\begin{pmatrix} a \sqsubset b \\ \textit{and} \\ x \sqsubset y \end{pmatrix} \quad \Rightarrow \quad (a \circ x) \sqsubset (b \circ y)$

In other words, a residuated semi-group is in particular an ordered semi-group.

Given a residuated semi-group, an ***interpretation*** $[\ldots]$ is a map from primitive types to elements in M, which extends to types and sequences of types in the obvious way:

$$\begin{array}{rclcrcl} [A, B] & = & [A] \circ [B] & \qquad & [A \setminus B] & = & [A] \setminus\!\!\setminus [B] \\ [A \bullet B] & = & [A] \circ [B] & \qquad & [B \mathbin{/} A] & = & [B] \mathbin{/\!/} [A] \end{array}$$

A sequent $\Gamma \vdash C$ is said to be ***valid*** in a residuated semi-group whenever $[\Gamma] \sqsubset [C]$.

The free group model

The free group interpretation for L is

- a particular residuated semi-group where
  - $(M, \cdot)$ is the free group over the propositional variables,
  - $a \mathbin{\backslash\!\backslash} b$ is $a^{-1}b$
  - $b \mathbin{/\!/} a$ is ba^{-1}
  - $a \sqsubseteq b$ is $a = b$ (the discrete order)
  - (RSG) holds: $ab = c \equiv a = cb^{-1} \equiv b = a^{-1}c$
- standard interpretation $[p] = p$

L sound w.r.t. residuated semi-groups (next proposition) hence $\Gamma \vdash C$ entails one has $[\Gamma] = [C]$ in the free group.

The free group model is of course not complete: indeed it interprets $\vdash$ by a symmetrical relation ($=$) while $\vdash$ is not symmetrical: $n \vdash s \,/\, (n \setminus s)$ is provable but not $s \,/\, (n \setminus s) \vdash n$.

This model will be especially important for converting Lambek grammars into Context-Free Grammars.

L is sound and complete w.r.t. residuated semi-groups

An easy induction yields:

Proposition 15 *A provable sequent is valid in every residuated semi-group, for every interpretation of the primitive types.*

A standard Lindenbaum algebra construction yields:

Proposition 16 *A sequent which is valid in every residuated semi-group is derivable.*

L is sound and complete w.r.t. (free) semi-group models

Much ore interesting: a category x is interpreted by the set of strings whose category is x.

Given a semi-group $(W, .)$ a set W with an associative composition "." one defines a residuated semi-group by:

- $M = 2^W$
- $A \circ B = \{ab \quad | \quad a \in A$ and $b \in B\}$
- $A \setminus\!\!\setminus B = \{z \quad | \quad \forall a \in A \quad az \in B\}$
- $B \,/\!/\, A = \{z \quad | \quad \forall a \in A \quad za \in B\}$
- $A \sqsubset B$ whenever $A \subset B$ (as sets).

It is easily seen that this structure really is a residuated semi-group:

- $\circ$ is associative:

  $$(A{\circ}B){\circ}C = \{abc \quad | \quad a \in A \text{ and } b \in B \text{ and } c \in C\} = A{\circ}(B{\circ}C)$$

- $\subset$ is an order on 2^W

(RSG) The following statements are clearly equivalent:

$$\begin{array}{rl} (A \circ B) \subset C & : \forall a \in A \, \forall b \in B \quad ab \in C \\ A \subset (C \,/\!\!/\, B) & : \forall a \in A \quad a \in (C \,/\!\!/\, B) \\ B \subset (A \,\backslash\!\!\backslash\, C) & : \forall b \in B \quad b \in (A \,\backslash\!\!\backslash\, C) \end{array}$$

What is of special interest are free semi-group models, since there are no equations between sequences of words. The following result may be understood as L is the logic of free semi-groups:

Proposition 17 *Product free L is complete over free semi-group models. So is L with product, but it is much more complicated to establish.*

PROOF : Take as semi-group the finite non empty sequences of formulae F^+, endowed with concatenation $(A_1, \ldots, A_n) \cdot (B_1, \ldots, B_p) = A_1, \ldots, A_n, B_1, \ldots, B_p$.

For a primitive type p define $[p]$ by $\{\Gamma \quad | \quad \Gamma \vdash p\}$.

Let us firstly see that for every formula F, the set of finite sequences $[F]$ defined inductively from the $[p]$'s by the definition of $\backslash\backslash$ and $//$ is precisely $Ctx(F) = \{\Delta \quad | \quad \Delta \vdash F\}$.

We proceed by induction on F. Is F if some primitive type, it is the definition. Now assume that $[G] = Ctx(G)$ and $[H] = Ctx(H)$ and let us see that $[G \setminus H] = Ctx(G \setminus H)$ — the case $H \mathbin{/} G$ being symmetrical.

$Ctx(G \setminus H) \subset [G \setminus H]$ Let Δ be a sequence such that $\Delta \in Ctx(G \setminus H)$ that is $\Delta \vdash G \setminus H$ (1) and let us see that for every $\Theta \in [G]$ we have $\Theta, \Delta \in [H]$ — which entails $\Delta \in [G \setminus H]$. By induction hypothesis we have $Ctx(G) = [G]$ so $\Theta \vdash G$ (2). From (1) and (2) we obtain $\Theta, \Delta \vdash H$, so $\Theta, \Delta \in Ctx(H)$. Since by induction hypothesis $Ctx(H) = [H]$ we have $\Theta, \Delta \in [H]$. As this holds for every Θ we have $\Delta \in [G \setminus H]$.

$[G \setminus H] \subset Ctx(G \setminus H)$ Let Δ be a sequence such that $\Delta \in [G \setminus H]$. Let us show that $\Delta \vdash G \setminus H$. Since $G \vdash G$ we have $G \in Ctx(G)$ and by induction hypothesis $G \in [G]$. By definition of $[G \setminus H]$ we thus have $G, \Delta \in [H]$ and, since by induction hypothesis we have $[H] = Ctx(H)$ we obtain $G, \Delta \vdash H$. Now, by the $\setminus_i$ introduction rule we obtain $\Delta \vdash G \setminus H$, that is $\Delta \in Ctx(G \setminus H)$.

If a sequent$A_1, \ldots, A_n \vdash C$ is valid in this model under this interpretation, what does it means? We have $[A_1] \circ \cdots \circ [A_n] \subset [C]$ and as $A_i \in [A_i]$ we have $A_1, \ldots, A_n \in [C]$ that is $A_1, \ldots, A_n \vdash C$. ◇

Lambek grammars
and context-free grammars:
Pentus’s proof of Chomsky conjecture

From context-free grammars to Lambek grammars

Surprisingly not completely straightforward... but not difficult.

We have seen that any AB grammar is weakly equivalent to an AB grammar only containing types of order at most 1 with categories
$p \qquad (p \,/\, q) \qquad ((p \,/\, q) \,/\, r)$.

Now, by Cohen's proposition a sequent $A_1, \ldots, A_n \vdash S$ with $o(A_i) \leq 1$ is provable with AB residuation rules only if and only if it is provable in L. Consequently the language generated by an AB grammar with types of order at most 1 coincide with the language generated by the Lambek grammar with the same lexicon.

Using the weak equivalence between AB grammars and context-free grammars (propositions 3 and 2) we have the result of :

Proposition 18 *Every ε-free context-free grammar is weakly equivalent to a Lambek grammar.*

A property of the free group

Let w be an element of the free group; then $\|w\|$ stands for the length of the reduced word corresponding w — e.g. $\|cb^{-1}a^{-1}abc\| = 2$.

Proved by Nivat 1971, Autebert Boasson Sénizergues 1984, Pentus 1992,...

Proposition 19 *The two following properties of the free group hold:*

1. *Let u, v, w be elements of the free group; if $\|u\| < \|uv\|$ and $\|uv\| \geq \|uvw\|$ then $\|vw\| \leq \max(\|v\|, \|w\|)$.*

2. *Let u_i $i = 1, \ldots, n+1$ be elements of the free group with $u_1 \cdot \cdots \cdot u_{n+1} = 1$. Then there exists $k \leq n$ such that*

$$\|u_k u_{k+1}\| \leq \max(\|u_k\|, \|u_{k+1}\|)$$

Interpolation for thin sequents

A sequent $\Gamma \vdash C$ is said to be ***thin*** whenever it is provable and $\rho_p(\Gamma, C)$ is at most 2 — where $\rho_p(\Theta)$ is the number of occurrences of a primitive type p in Θ.

In a thin sequent $\Gamma \vdash C$ the number $\rho_p(\Gamma, C)$ is either 0 or 2.

Here is a proposition which is very representative of multiplicative calculi, in which a formula is neither contracted or weakened:

Proposition 20 *Each provable sequent may be obtained from a thin sequent by substituting primitive types with primitive types.*

PROOF : Given a cut free proof d with only primitive axioms of a sequent $\Gamma \vdash C$, number the axioms and replace each axiom $p \vdash p$ by $p_i \vdash p_i$ where i is the number of the axiom, and also replace all the traces of this occurrence of p in the proof with p_i. Clearly the result is itself a proof of a sequent $\Gamma' \vdash C'$, which contains exactly two or zero occurrences of each primitive type, and which gives back $\Gamma \vdash C$ when each p_i is substituted with p. $\diamond$

Proposition 21 *Let $\Gamma, \Delta, \Theta \vdash C$ be a thin sequent. Then there exists a formula B such that:*

1. *$\Delta \vdash B$ is thin*
2. *$\Gamma, B, \Theta \vdash C$ is thin*
3. *$|B| = \|[\Delta]\|$ — the number of primitive types in B is the size of the interpretation of Δ in the free group.*

Proposition 22 *Let $A_1, \ldots, A_n \vdash A_{n+1}$ be a thin sequent with $|A_i| \leq m$; then either:*

- *there exists an index k and a type B with $|B| \leq m$ such that the following sequents are thin:*

  $$A_1, \ldots, A_{k-1}, B, A_{k+2}, \ldots, A_n \vdash A_{n+1}$$
  $$A_k, A_{k+1} \vdash B$$

- *there exist a type B with $|B| \leq m$ such that the following sequents are thin:*

  $$B, A_n \vdash A_{n+1}$$
  $$A_1, \ldots, A_{n-1} \vdash B$$

From Lambek grammars to context-free grammars

Proposition 23 *If a sequent* $A_1, \ldots, A_n \vdash A_{n+1}$ *with each* $|A_i| \leq m$ *is provable in L, then it is provable from provable sequents* $U, V \vdash X$ *or* $U \vdash X$ *with* $|U|, |V|, |X| \leq m$ *by means of the cut rule only.*

PROOF : We proceed by induction on n. If $n \leq 2$ then there is nothing to prove. Otherwise, let $A'_1, \ldots, A'_n \vdash A'_{n+1}$ be a corresponding thin sequent obtained as in proposition 20 — using a different primitive type for each axiom in the proof of $A_1, \ldots, A_n \vdash A_{n+1}$. Thus there exists a substitution σ replacing primitive types with primitive types and preserving provability such that $\sigma(A') = A$.

As the substitution replaces primitive types with primitive types, we also have $|A'_i| \leq m$. By proposition 22 there exists a formula B' with $|B| \leq m$ such that either:

- $A'_1, \ldots, A'_{k-1}, B', A'_{k+2}, \ldots, A'_n \vdash A'_{n+1}$
  $A'_k, A'_{k+2} \vdash B'$
  are thin, and therefore provable. Let $B = \sigma(B')$, so B has at most m primitive types as well; applying the substitution we obtain two provable sequents
  $A_1, \ldots, A_{k-1}, B, A_{k+2}, \ldots, A_n \vdash A_{n+1}$
  $A_k, A_{k+1} \vdash B$.
  By induction hypothesis
  $$A_1, \ldots, A_{k-1}, B, A_{k+2}, \ldots, A_n \vdash A_{n+1} \qquad (*)$$
  is provable from provable sequents $U, V \vdash X$ or $U \vdash X$ with $|U|, |V|, |X| \leq m$ by means of the cut rule only.
  Notice that $A_k, A_{k+1} \vdash B \quad (**)$ is of the form $U, V \vdash X$ with $|U|, |V|, |X| \leq m$.
  A cut rule between the proof of $(*)$ and $(**)$ yields a proof of
  $$A_1, \ldots, A_n \vdash A_{n+1}$$
  from provable sequents $U, V \vdash X$ or $U \vdash X$ with $|U|, |V|, |X| \leq m$ by means of the cut rule only.

- $B', A'_n \vdash A'_{n+1}$ and $A_1, \ldots, A_{n-1} \vdash B$ are thin and therefore provable. Let $B = \sigma(B')$, so $|B| \leq m$; applying the substitution we obtain two provable sequents

  $B, A_n \vdash A_{n+1}$
  $A_1, \ldots, A_{n-1} \vdash B$.

  By induction hypothesis

  $$A_1, \ldots, A_{n-1}, B \vdash A_{n+1} \qquad (+)$$

  is provable from provable sequents $U, V \vdash X$ or $U \vdash X$ with U, V, X having at most m primitive types by means of the cut rule only. Notice that $B, A_n \vdash A_{n+1}$ $(++)$ is of the form $U, V \vdash X$ with $|U|, |V|, |X| \leq m$.
  A cut rule between the proof of $(+)$ and $(++)$ yields a proof of

  $$A_1, \ldots, A_n \vdash A_{n+1}$$

  from provable sequents $U, V \vdash X$ or $U \vdash X$ with $|U|, |V|, |X| \leq m$ by means of the cut rule only.

◇

Theorem 24 *Let Lex be the lexicon of a Lambek grammar G_L, and let and let m the maximal number of primitive types in a formula of the lexicon. Then the language $L(G_L)$ generated by G_L is the same as the language $L(G_C)$ generated by the following context-free grammar G_C:*

- *Terminals: terminals (words) of G_L*
- *Non-Terminals: all formulae A with $|A| \leq m$*
- *Start symbol S, the one of G_L*
- $X \longrightarrow a$ *whenever* $X \in \mathrm{Lex}(a)$
- $X \longrightarrow A$ *whenever* $A \vdash X$ *is provable in* L
- $X \longrightarrow A\,B$ *whenever* $A, B \vdash X$ *is provable in* L

Observe that the rules are in finite number, because there are finitely many sequents $U, V \vdash X$ or $U \vdash X$ when U, V, X contains at most m primitive types — hence there are only finitely many provable such sequents.

PROOF : Assume $a_1 \cdots a_n \in L(G_C)$. Hence there exist types $X_i \in \mathrm{Lex}(a_i)$ such that $S \longrightarrow X_1 \cdots X_n$. The derivation in the CFG G_C can be turned into a derivation in L using only the cut rule (reversing $\longrightarrow$ and $\vdash$), therefore $a_1 \cdots a_n \in L(G_L)$.

Assume now that $a_1 \cdots a_n \in L(G_L)$. Hence there exist types $X_i \in \mathrm{Lex}(a_i)$ such that $X_1, \ldots, X_n \vdash S$. By proposition 23 such a sequent is provable by means of the sequents corresponding to production rules, and of the cut rule only.

By induction on the size of the cut-only proof, it is easily seen that the proof corresponds to a derivation in the CFG G_C.

If the proof is reduced to a proper axiom, than this axiom is itself a production rule.

If the last rule is a cut, say between $\Gamma, B, \Theta \vdash C$ and $\Delta \vdash B$, then by induction hypothesis we have $B \longrightarrow \Delta$ and $C \longrightarrow \Gamma\, B\, \Theta$ hence $C \longrightarrow \Gamma\, \Delta\, \Theta$. Thus, if $a_1 \cdots a_n \in L(G_L)$, we have $S \longrightarrow X_1 \cdots X_n$ with $X_i \in \mathrm{Lex}(A_i)$; as $X_i \in \mathrm{Lex}(a_i)$ we have $S \longrightarrow a_1 \cdots a_n$. ◇

Comments

Pentus results does not mean that Lambek grammars are useless.

Firstly it is a weak equivalence (yielding different syntactic structures)

Secondly, context-free grammars have a very different structure:

- language specific rules
- non structured (meaningless) categories (non terminals)
- learning complicated (not lexicalized, at least with good linguistic structures)
- correspondence with semantics more difficult

This result suggests to look for extension beyond context-free ness which keeps this deductive framework in particular for computing semantic representation. (e.g. Multi Modal Categorial Grammars, Categorial Minimalist Grammars)

Outcomes of this restricted CGs

Here we have only seen the simplest logical system for grammar as a deductive system, a purely logical one: Lambek calculus and grammars.

A grammar system quite different from standard formal language theory. (Evidence for the difference: the proofs are already in HR graph/tree grammars, incomparable with context free tree languages).

Because of its relation to ordinary logic, in particular intuitionistic logic, perfect for computing logical semantic representation in a way that impements compositionality).

Because of the lexicalisation and of the structure of the deductions/parse structure, easy to learn from structure, with a convergent algorithm.

Discussion

Pretty mathematical results from formal linguistics motivation.

Assuming the linguistic relevance of the model, are the mathematical results used?

- The connection with usual logic is clearly used especially for computing semantic representations.
- The existence of normal proof is used every where, but not the process of normalisation.
- Free semi group model is appealing, but not really used.
- Learning technique could be used for grammar construction from corpora (e.g. also works for Lambek grammars Bonato, Retoré)

Discussion

Is the linguistic model relevant?

The implicit claim is that deduction in a resource logic is related to linguistic processing and for some of us that the first includes the second.

- Orthodox view of deduction as parsing Multi Modal Categorial Grammars (Moortgat) Various connectives assosciative or not, commutative or not, modality and postulates for relation among them.

- Valency consumption(for semantic applciation) by commutative linear logic Intermediate out and word order as a separate process:
  - completely separate (IG Perrier, ACG de Groote)
  - partially separate, with the connectives commutative and not commutative à la de Groote Abrusci Ruet (CMG Lecomte, Retoré)

References

Two-part survey in the journal *La Gazette des mathématiciens* (Société Mathématique de France).
Ch. Retoré Les mathématiques de la linguistique computationnelle.
Premier volet: la théorie des langages. Volume 115 janvier 2008 pp. 35-62.
`http://smf.emath.fr/Publications/Gazette/2008/115/`
Second volet: Logique. Volume 116 avril 2008 pp. 29-63.
`http://smf.emath.fr/Publications/Gazette/2008/116/`

Lecture notes Ch. Retoré The logic of categorial grammars - Lecture Notes - Rapport de Recherche INRIA 5703
`http://www.inria.fr/rrrt/rr-5703.html`