Toward Discourse Representation Via Pregroup Grammars

Anne Preller

Toward Discourse Representation via Pregroup Grammars

to appear in: JoLLI, Springer, 2007

Anne Preller
Lirmm, Montpellier, France
preller@lirmm.fr

Abstract

Every pregroup grammar is shown to be strongly equivalent to one which uses basic types and left and right adjoints of basic types only. Therefore a semantical interpretation is independent of the order of the associated logic. Lexical entries are read as expressions in a two sorted predicate logic with $\in$ and functional symbols. The parsing of a sentence defines a substitution that combines the expressions associated to the individual words. The resulting variable free formula is the translation of the sentence. It can be computed in time proportional to the parsing structure. Non-logical axioms are associated to certain words (relative pronouns, indefinite article, comparative determiners). Sample sentences are used to derive the characterizing formula of the DRS corresponding to the translation.

Keywords: categorial grammars, pregroup grammars, Discourse Representation, semantic interpretation

1 Introduction

Pregroups are introduced in [Lambek 1999] as a simplification of his syntactic calculus [Lambek 1958] and belong to the group of categorial grammars. Like these they use proofs in a formal system to derive sentences, but unlike these, they do not have a canonical interpretation in Montague semantics. Therefore the slogan characterizing categorial grammars, namely that 'all the grammar is in the dictionary', seems no longer to apply, at least if one accepts the imperative that a grammar should handle both syntax and semantics.

Below, *standard* pregroup dictionaries are defined. They involve the so-called first order fragment of free pregroups and have a natural interpretation in predicate logic. However, due to the Grishin Equalities, a distinction between first order and higher order serves no purpose in pregroup grammars: Indeed, the main theorem of this paper shows that all pregroup grammars are strongly equivalent to standard grammars. Here, strongly equivalent means not only that

the grammars generate the same sentences but also that a parsing of a sentence in one grammar can be transformed into a parsing in the other grammar. In fact, the geometrical structure of the two parsings remains the same, only the labels of the nodes change. This geometrical structure, called reduction below, is a simple planar graph formed by the under-links one draws to indicate that the (generalized contraction) rule of the pregroup grammar applies. Reductions correspond to the parsing trees of rewrite-grammars or the lambda-terms coding proofs in categorial grammars. Hence a *parsing* of a sentence consists now of a choice of types from the dictionary and a reduction to the sentence type.

Adding semantics, we extend the parsings of sentences to structures which are close in concept to the situation schemata of [Fenstad et al.]. The main idea is to assign logical expressions to words and then compute the translation of the sentence from that of the words. Hence we define

- a *translation* map that assigns to each lexical entry one or more functional symbols. Non-logical axioms are attached to some of these symbols.
- an *interpretation* map which assigns to each parsing a variable-free expression and instances of the non-logical axioms, both obtained by substitution according to the under-links of the parsing.

The translation map is given in the dictionary. The interpretation map is computable in time proportional to the reduction of the sentence. Such a reduction can be computed by an algorithm which runs in time proportional to the cube of the length n of the sentence. For example, [Degeilh-Preller] adapts [Earley]'s parsing algorithm to pregroup dictionaries obtaining a constant of n^3 that only depends on the maximal length of types and the maximal number of types per word in the dictionary. The dictionary need not be finite. Hence pregroup grammars have an efficient cubic time algorithm which recognizes, parses and interprets sentences by logic formulas.

First we recall the basic definitions including the geometrical notion of reductions. Then we present the semantics, illustrating its working by a standard dictionary. The sentences, given both in French and in English, cover existential and universal quantifiers and long distance dependencies related to the relative pronoun. In the next section, we prove the Strong Equivalence Theorem, after introducing the main tool, the notion of complexity of a set of types. Complexity is weaker than the notion of order in **AB**-grammars, a notion which can also be defined in pregroup grammars. We show that every pregroup dictionary of finite complexity, and therefore every pregroup grammar in the sense of [Buszkowski] is strongly equivalent to a standard dictionary. Finally in Section 5, we use this theorem to define a semantically meaningful interpretation of the long distance dependences caused by the comparative.

2 Geometry of derivations

We briefly recall the definition of pregroups and the construction of a freely generated pregroup defined in [Lambek 1999]. As we are interested in parsing

and semantical interpretation, we look at the actual derivations in a free pregroup, concentrating on derivations that consist of generalized contractions only. The geometrical structure of these derivations is at the base of the semantical interpretation in the following Section 3.

A preordered monoid $< P, 1, \cdot, \rightarrow >$ is a set with at least one element $1 \in P$, a binary operation $\cdot$ and a binary relation $\rightarrow$ satisfying for all $a, b, c, u, v \in P$

$1 \cdot a = a = a \cdot 1$

$(a \cdot b \cdot)c = a \cdot (b \cdot c)$

$a \rightarrow a$

$a \rightarrow b$ and $b \rightarrow c$ imply $a \rightarrow c$

$a \rightarrow b$ implies $u \cdot a \cdot v \rightarrow u \cdot b \cdot v$.

The dot denotes multiplication and is generally omitted. The arrow $\rightarrow$ denotes the preorder.

A pregroup is a partially preordered monoid in which each element a has both a *left adjoint* a^ℓ and a *right adjoint* a^r satisfying

(Contraction) $a^\ell a \rightarrow 1, \ aa^r \rightarrow 1$

(Expansion) $\quad 1 \rightarrow a^r a, \ 1 \rightarrow aa^\ell$.

One derives

1. $a \rightarrow b$ if and only if $b^\ell \rightarrow a^\ell$ if and only if $b^r \rightarrow a^r$,
2. $a \rightarrow b$ if and only if $ab^r \rightarrow 1$ if and only if $b^\ell a \rightarrow 1$.

The *free* pregroup $P(B)$ generated by a partially ordered set of *basic types* B is characterized in [Lambek 1999] as the preordered free monoid generated from the set of *simple types* Σ consisting of the basic types and their *iterated adjoints*

$$\Sigma = \left\{ a^{(z)} : a \in B, z \in \mathbb{Z} \right\} .$$

The elements of $P(B)$ are called *types*, they are strings of the form

$a_1^{(z_1)} \dots a_k^{(z_k)}$,

where $a_1, \dots, a_k$ are basic types and $z_1, ..., z_k$ are integers. The unit 1 denotes the empty string and multiplication is the same as concatenation.

The left and right adjoints of a type are defined by

$(a_1^{(z_1)} \dots a_k^{(z_k)})^\ell = a_k^{(z_k - 1)} \dots a_1^{(z_1 - 1)}$

$(a_1^{(z_1)} \dots a_k^{(z_k)})^r = a_k^{(z_k + 1)} \dots a_1^{(z_1 + 1)}$.

Hence, identifying the basic type $a \in B$ with $a^{(0)} \in \Sigma$ we have

$a^{\ell\ell} = a^{(-2)}, a^\ell = a^{(-1)}, a = a^{(0)}, a^r = a^{(1)}, a^{rr} = a^{(2)}$ etc.

If $s = a^{(z)}$ we call z the *iterator* of s.

Finally, the preorder on types is defined as the transitive closure of the union of the following three relations

$$\begin{array}{ll} \text{(Induced step)} & Xa^{(z)}Y \rightarrow Xb^{(z)}Y \\ \text{(Generalized contraction)} & Xa^{(z)}b^{(z+1)}Y \rightarrow XY \\ \text{(Generalized expansion)} & Y \rightarrow Xa^{(z+1)}b^{(z)}Y \end{array} ,$$

where X and Y are arbitrary types, a and b are basic and either z is even and $a \to b$ or z is odd and $b \to a$.

The property which makes the theory of pregroups decidable is expressed in the so-called

Switching Lemma (Proposition 2 of [Lambek 1999]):
Let $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ be simple types. Then $a_1 \ldots a_n \to b_1 \ldots b_m$ if and only if there are a substring $a_{i_1} \ldots a_{i_k}$ of $a_1 \ldots a_n$ and a substring $b_{i_1} \ldots b_{i_k}$ of $b_1 \ldots b_m$ such that

$$a_1 \ldots a_n \to a_{i_1} \ldots a_{i_k} \to b_{i_1} \ldots b_{i_k} \to b_1 \ldots b_m,$$

$$a_{i_p} \to b_{i_p}, \text{for } 1 \leq p \leq k,$$

where $a_{i_1} \ldots a_{i_k}$ is obtained from $a_1 \ldots a_n$ by generalized contractions only, $b_1 \ldots b_m$ is obtained from $b_{i_1} \ldots b_{i_k}$ by generalized expansions only and $b_{i_1} \ldots b_{i_k}$ is obtained from $a_{i_1} \ldots a_{i_k}$ by induced steps only.

In linguistic applications, the relevant inequalities have the form $s_1 \ldots s_m \to s$, where the s_i's are simple and s is a basic type. A derivation of such an inequality can be obtained by generalized contractions and induced steps only. For example, consider the dictionary

$$\begin{array}{lcl} \textit{Marie} & : & \nu \\ \textit{Jean} & : & \nu \\ \textit{adore} & : & \pi^r s o^\ell \end{array}$$

The basic types ν, π, o and s stand for 'proper name', 'subject third person singular'[1], 'object' and 'sentence in the present' respectively. They satisfy $\nu \to o$ and $\nu \to \pi$. To analyze the French sentence

Marie adore Jean (Mary adores John)

concatenate the types from the dictionary in the order of the words. The obtained type has a derivation to the sentence type

$$\begin{array}{lll} \textit{Marie} & \textit{adore} & \textit{Jean} \\ \underbrace{(\nu)} & \underbrace{(\pi^r \; s \; o^\ell)} & \underbrace{(\nu)} \end{array} \quad \to s$$

This derivation is justified by the generalized contractions $\nu\pi^r \to 1$ and $o^\ell\nu \to 1$. As customary, the types have been written under the words and the generalized contractions are indicated by under-links.

By Property 2. above, every derivation of $s_1 \ldots s_n \to s$ is equivalent to a derivation of $s_1 \ldots s_n s^r \to 1$. In the case of the example above, the previously unlinked s is now linked to s^r:

$$\underbrace{\nu \quad \pi^r} \; \underbrace{s \; o^\ell \quad \nu \quad s^r}\,.$$

In fact, the under-links uniquely determine the derivation to the empty string. We decompose such a derivation into a geometrical part called *reduction*, consisting of the set of under-links, and an algebraic part, consisting of generalized contractions. Following [Preller-Lambek], such geometrical representations of derivations will be called *transitions*.

[1] We ignore agreement in person, number, gender etc. to keep exposition simple.

Definition 1 (**Reduction**)
A set R consisting of sets $\{i,k\}$, $1 \leq i \neq k \leq n$, is a *reduction with respect to* n if the following two *geometrical* conditions hold

1) for every i with $1 \leq i \leq n$, there is exactly one k such that $\{i,k\} \in R$,

2) if $\{i,k\}$, $\{l,m\} \in R$ and $i < l < k$, then $i < m < k$.

A reduction R with respect to n and a string of simple types $s_1 \dots s_n$ are said to form a *transition of* $s_1 \dots s_n$ *to the empty string* 1, in symbols $R : s_1 \dots s_n \Rightarrow 1$, if the following *algebraic* condition is satisfied

3) $s_i s_k \to 1$, whenever $\{i,k\} \in R$ and $i < k$.

The element $\{i,k\} \in R$ is represented graphically by the under-link

$$\dots \underbrace{s_i \dots s_k} \dots, \text{ where } i < k.$$

Thus, a reduction R is a planar graph where the integers from 1 to n form the linearly ordered set of vertices. It satisfies the geometrical conditions, which can be reformulated thus

- there are no loops
- every i is endpoint of exactly one under-link,
- under-links do not cross.

The same reduction may constitute a transition to the empty string for quite different strings of simple types. For example, the transitions

$$\underbrace{a^r \; \underbrace{a^\ell \; \underbrace{a^\ell \; a} \; a} \; a^{rr}} \to 1\,, \quad \underbrace{b \; \underbrace{b^{\ell\ell} \; \underbrace{b^{\ell\ell} b^\ell} \; b^\ell} \; b^r} \to 1\,, \quad \underbrace{c \; \underbrace{c \; \underbrace{c \, c^r} \; c^r} \; c^r} \to 1$$

have the same links, namely $\{1,6\}$, $\{2,5\}$ and $\{3,4\}$. Hence their geometrical structures are identical. We say that transitions $R : s_1 \dots s_n \Rightarrow 1$ and $R' : s'_1 \dots s'_{n'} \Rightarrow 1$ are *similar* if $R = R'$. This implies in particular that $n = n'$, but of course in general $s_i \neq s'_i$. Clearly, the isomorphic image of a transition is similar to the original transition. However, similarity is weaker than isomorphism. Indeed, isomorphism is a global notion whereas similarity is local. If a is an isolated element we can define an isomorphism that maps a to a^r and hence $aa^{\ell\ell}a^{\ell\ell}a^\ell a^\ell a^r$ to $a^r a^\ell a^\ell aaa^{rr}$, but no isomorphism would ever map any of these two types to $cccc^r c^r c^r$.

Note that if $R : s_1 \dots s_n \Rightarrow 1$ is a transition then the iterator of a right endpoint of a link is the successor of the iterator of its left endpoint: Indeed, if $\{i,k\} \in R$ and $i < k$, then the algebraic condition $s_i s_k \to 1$ implies that $s_i = a^{(z)}$ and $s_k = b^{(z+1)}$ for some integer z and appropriate basic types a and b.

3 Interpretation of sentences in predicate logic

The interpretation of the sample sentences below is presented in the style of discourse representation in [Kamp-Reyle]. We use two-sorted predicate logic, one sort for individuals and the other one for sets of individuals. This two-sorted approach is motivated by the handling of singulars and plurals in a way similar to natural language. It has two primitive relational symbols, namely $\in$ and $=$. The former requires individuals on the left and sets on the right, the latter accepts either sets on both sides or individuals on both sides. We refer to an individual or a set as an *entity*. Variables are denoted by lower case letters $x, y, \ldots$ and range over entities. Hence in the formula $x \in y$, the variable x stands for an individual and y for a set. Moreover, there are a set constant Ω, the *set of truth values*, and individual constants $\top \in \Omega$, $\bot \in \Omega$. The specific properties of Ω my be specified as usual, they do not matter for this introduction. The symbols mentioned so far are called *logical*.

To the logical symbols are added *non-logical symbols*, which are functional symbols, defined by the entries in the dictionary. They come with an *arity* and a sort. The former indicates the number of argument places of the interpreting function, the latter is the sort of its values. Each argument place can be occupied indifferently by a set or an individual. Among the non-logical individual symbols we distinguish the *predicate symbols* for which the interpreting function takes its values in Ω. Expressions are defined from the variables and the non-logical symbols by induction as usual, hence they have no occurrences of $=$ or $\in$. If the dominating symbol is a predicate symbol, we call the expression a *predicate expression*. The other expressions are the *entity expressions*. For example, the sentences MARY LIKES BOOKS and JOHN LIKES MARY add individual constants `mary`, `john`, a set constant `Book` and a binary predicate symbol `like` to the logic. The *translation* of a sentence is a variable free predicate expression, for example, `like(mary,Book)` and `like(john,mary)`. This translation with its tree-like structure replaces the D(iscourse) R(epresentation) S(tructure) of [Kamp-Reyle].

Instead of conditions under which a DRS is true, we have non-logical axioms associated to words, explaining their meaning. The truth of a predicate expression p is expressed by the equality $p = \top$. To increase readability, this atomic formula is replaced by the expression p alone, if the context permits. For example, $\forall z(z \in X \Rightarrow p(z))$ stands for $\forall z(z \in X \Rightarrow p(z) = \top)$. The instances of the non-logical axioms together with the translation constitute the *interpretation* of the sentence. They imply the variable free first order formula associated to the DRS in [Kamp-Reyle]. The reason why the DRS formula is sometimes weaker than the translation is that we treat quantifiers as functions from sets to sets.

The first non-logical axiom explains the meaning of count-nouns. For example,

$$\texttt{like}(\texttt{mary}, \texttt{Book}) \Leftrightarrow \forall z(z \in \texttt{Book} \Rightarrow \texttt{like}(\texttt{mary}, z))$$

says that Mary likes the set of books if and only if she likes every book. We can safely assume that this property is valid for all predicate and set expressions

of the language, provided the new variable does not occur within the scope of proper subexpressions. Hence we say that the variable z is *accessible* in the predicate expression p if for every subexpression p' of p, either z has no occurrence in p' or at least one of its occurrences in p' is not inside a proper subexpression of p'.

Non-logical Axiom

(1) $p(X) \Leftrightarrow \forall z(z \in X \Rightarrow p(z))\,,$

where X is a set expression, p a predicate expression and z is accessible in p.

The interpretation of sentences via pregroup grammars can be computed by an algorithm. Here, we illustrate its working with a few examples where we ignore the dependence on person, number and gender of noun phrases, verbs etc., for simplicity's sake. Consider the sentence

Jean aime Marie (John likes Mary)

and its usual rendering in predicate logic, the atomic formula

`aimer(jean,marie)`

where `aimer` is a binary relational symbol and `jean` and `marie` are individual constants. We represent this correspondence by

$$\begin{array}{lcll} \textit{Jean} & : & \nu & \texttt{jean} \\ \textit{Marie} & : & \nu & \texttt{marie} \\ \textit{aime} & : & \pi^r s o^\ell & \texttt{aimer}(x_1, x_2) \end{array} .$$

In the last item, x_1 and x_2 are variables or argument places. By convention, x_1 corresponds to the first non-basic type, here π^r, and x_2 to the second non-basic type, here o^ℓ. We assume

$\nu \to \pi$ and $\nu \to o$.

The corresponding translation,

$\texttt{aimer(jean,marie)} = \texttt{aimer}(x_1, x_2)[x_1, x_2|\texttt{jean}, \texttt{marie}]$

can be computed by substituting the constants at the first and second argument place of the binary symbol. This substitution follows the links of the reduction

$$\begin{array}{l} \textit{Jean aime} \quad \textit{Marie} \\ \underbrace{(\nu)}\ \underbrace{(\pi^r}\ \underbrace{s\ o^\ell)}\ \underbrace{(\nu)}\ \underbrace{\qquad s^r}\ . \end{array}$$

We remark that

- each basic type corresponds to a functional symbol,
- each non-basic type corresponds to an argument place
- a transition to the sentence type defines a translation of the string of words into a variable free expression. The links indicate the argument place where the translated expressions are substituted.

Consider another, slightly more complicated example:

$$\begin{array}{lcll} \textit{un} & : & \pi c^\ell & \texttt{un}(y) \\ \textit{homme} & : & c & \texttt{Homme} \\ \textit{aime} & : & \pi^r s o^\ell & \texttt{aimer}(x_1, x_2) \\ \textit{Marie} & : & \nu & \texttt{marie} \end{array}$$

where c is the type for count nouns.

The translation of $un : \pi c^\ell$ is a unary individual functional symbol. The count noun is translated by a set constant. Using the links of the reduction

Un *homme* *aime* *Marie*
A MAN LIKES MARY
$(\pi\, c^\ell)\ (c)\ (\pi^r\ s\, o^\ell)\ (\nu)\quad s^r$,

we compute the translation by substituting

$\texttt{aimer}(x_1, x_2)\,[x_1, x_2|\texttt{un}(y)\,[y|\texttt{Homme}]\,, \texttt{marie}]$
$\equiv \texttt{aimer}(x_1, x_2)\,[x_1 x_2|\texttt{un}(\texttt{Homme}), \texttt{marie}]$
$\equiv \texttt{aimer}(\texttt{un}(\texttt{Homme}), \texttt{marie}))$.

However, finding the translation of the sentence is not enough. We have to express our understanding of the words by non-logical axioms. For example, the indefinite article *un* is interpreted as a unary functional symbol, which a model must realize as a function that maps a set of individuals to an individual in the set. Hence the

Non-logical Axiom

(2) $\texttt{un}(X) \in X$, where X ranges over set expressions.

Then we have the following instance of the non logical axiom

$\texttt{un}(\texttt{Homme}) \in \texttt{Homme}$.

Introducing a new constant

$\texttt{c} = \texttt{un}(\texttt{Homme})$,

we derive from the translation

$\texttt{aimer}(\texttt{un}(\texttt{Homme}), \texttt{marie}))$

the following formulas

$\texttt{aimer}(\texttt{c}, \texttt{marie})$, $\texttt{c} \in \texttt{Homme}$, $\exists x(x \in \texttt{Homme} \wedge \texttt{aimer}(x, \texttt{marie}))$.

The latter is the first order formula chosen by [Kamp-Reyle] for the Discourse Representation Structure representing similar sentences.

c marie
c ∈ Homme
aimer(c, marie)

$\exists x(x \in \texttt{Homme} \wedge \texttt{aimer}(x, \texttt{mary}))$.

Not much changes, if the indefinite article *un* (A) is replaced by the determiner *chaque* (EACH, EVERY):

Jean	:	ν	$\texttt{jean}$
achète	:	$\pi^r s o^\ell$	$\texttt{acheter}(x_1, x_2)$
chaque	:	nc^ℓ	$\texttt{chaque}(y)$
livre	:	c	$\texttt{Livre}$,

where $n \to \pi$ and $n \to o$. Note that the lexical entry $chaque : nc^\ell$ is again translated by a unary functional symbol. The non-logical axiom accompanying the entry will give it the appropriate meaning.

Non-logical Axiom

(3) $\texttt{chaque}(X) = X$, where X ranges over set expressions.

This axiom expresses that a function selecting every individual of the set selects the whole set.

The reduction of the sentence

Jean achète chaque livre
JOHN BUYS EVERY BOOK
$\underbrace{(\nu}\ \ \underbrace{(\pi^r}\ \ s\ \underbrace{o^\ell)}\ \ \underbrace{(n}\ \ \underbrace{c^\ell)}\ \ \underbrace{(c}\ ) \to s$,

produces the translation

$\texttt{acheter}(\texttt{jean}, \texttt{chaque}(\texttt{Livre}))$.

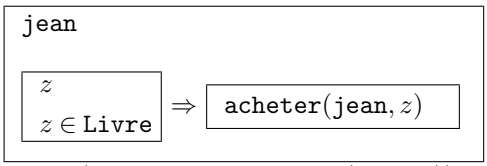
The instances of the non-logical axioms are

$\texttt{chaque}(\texttt{Livre}) = \texttt{Livre}$
$\texttt{acheter}(\texttt{jean}, \texttt{Livre}) \Leftrightarrow \forall z(z \in \texttt{Livre} \Rightarrow \texttt{acheter}(\texttt{jean}, z))$.

From these and the translation of the sentence we derive

$\texttt{acheter}(\texttt{jean}, \texttt{Livre})$
$\forall z(z \in \texttt{Livre} \Rightarrow \texttt{acheter}(\texttt{jean}, z))$.

Compare this with the corresponding DRS and its characterizing first order formula



$\forall z(z \in \texttt{Livre} \Rightarrow \texttt{acheter}(\texttt{jean}, z))$

Note that the characterizing formula of the DRS and the non-logical axioms imply our translation.

If a noun phrase formed with the indefinite article is not in subject position, the selected individual in general depends on the preceding subject, for example *Chaque homme achète un livre* (EVERY MAN BUYS A BOOK) or *Jean achète un livre* (JOHN BUYS A BOOK). The answer to this difference in meaning is to add new types for the verb and the indefinite article:

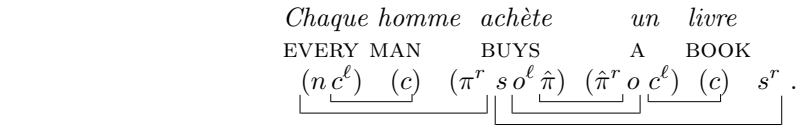
achète :	$\pi^r s o^\ell \hat{\pi}$	$\texttt{acheter}(x_1, x_2)$	$\texttt{id}(x_1)$
un :	$\hat{\pi}^r o c^\ell$	$\texttt{un}(x, y)$	

where $\hat{\pi}$ is a new isolated basic type. The new functional symbols are subject to the

Non-logical axioms

(4) $\texttt{id}(E) = E$, E arbitrary expression
(5) $x \in X \Rightarrow \texttt{un}(x, Y) \in Y$, X, Y set expressions.

For this choice of types, the reduction of the sentence is



The translation defined by this reductions is

$\texttt{acheter}(\texttt{chaque}(\texttt{Homme}), \texttt{un}(\texttt{id}(\texttt{chaque}(\texttt{Homme})), \texttt{Livre}))$.

Using the non-logical axioms we derive

$\texttt{acheter}(\texttt{Homme}, \texttt{un}(\texttt{Homme}, \texttt{Livre}))$
$z \in \texttt{Homme} \Rightarrow \texttt{acheter}(z, \texttt{un}(z, \texttt{Livre}))$
$z \in \texttt{Homme} \Rightarrow \texttt{un}(z, \texttt{Livre}) \in \texttt{Livre}$
$z \in \texttt{Homme} \Rightarrow \texttt{un}(z, \texttt{Livre}) \in \texttt{Livre} \wedge \texttt{acheter}(z, \texttt{un}(z, \texttt{Livre}))$
$\forall z(z \in \texttt{Homme} \Rightarrow \exists y(y \in \texttt{Livre} \wedge \texttt{acheter}(z, y)))$.

Our next example extends the noun phrases by restriction to relative clauses. Consider the dictionary

Jean	:	ν	$\texttt{jean}$
aime	:	$\pi^r s o^\ell \hat{\pi}$	$\texttt{aimer}(x_1, x_2) \quad \texttt{id}(x_1)$
un	:	$\hat{\pi}^r o c^\ell$	$\texttt{un}(x, y)$
livre	:	c	$\texttt{Livre}$
que	:	$c^r c \hat{o} \hat{s}^\ell$	$\texttt{Que}(y_1, y_2)$, $\texttt{Dummy}$
Marie	:	ν	$\texttt{marie}$
déteste	:	$\pi^r \hat{s} \hat{o}^r$	$\texttt{détester}(z_1, z_2)$.

Here we added new basic types $\hat{s}$ and $\hat{o}$. The latter differs from the type for pseudo-objects introduced in [Lambek 2004], denoted by the same symbol, because we assume $\hat{o}$ to be isolated in the set of basic types $B = \{\nu, \pi, n, c, o, \hat{o}, s, \hat{s}\}$, where only $\nu \to \pi$, $n \to \pi$, $\nu \to o$ and $n \to o$ hold. Note the new type for the transitive verb *déteste* which uses the sentence type $\hat{s}$ for relative clauses.

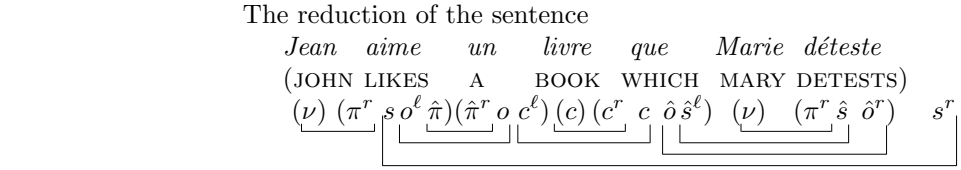
Two of the simple types in the string $c^r c \hat{o} \hat{s}^\ell$ are basic, namely c and $\hat{o}$. The translation therefore has two non-logical symbols, the binary functional set symbol $\texttt{Que}$ and the set constant $\texttt{Dummy}$. The symbol $\texttt{Que}$ corresponds to the basic type c, its argument places y_1 and y_2 correspond to the non basic types c^r and $\hat{s}^\ell$ in that order. The other basic type $\hat{o}$ is translated by the constant $\texttt{Dummy}$. This reflects our understanding that the relative pronoun selects among the individuals characterized by the preceding noun those which satisfy the predicate expressed by the following relative clause. Hence the

Non-logical axiom

(6) $\quad z \in \texttt{Que}(X, p(\texttt{Dummy})) \Leftrightarrow z \in X \wedge p(z)$,

where X ranges over set expressions, $p(z)$ over predicate expressions and z is accessible in p.

The reduction of the sentence



defines the translation

$\texttt{aimer}(\texttt{jean}, \texttt{un}(\texttt{id}(\texttt{jean}), \texttt{Que}(\texttt{Livre}, \texttt{détester}(\texttt{marie}, \texttt{Dummy}))))$.

Assuming $\texttt{c} = \texttt{un}(\texttt{id}(\texttt{jean}), \texttt{Que}(\texttt{Livre}, \texttt{détester}(\texttt{marie}, \texttt{Dummy})))$, we derive

$\texttt{c} \in \texttt{Que}(\texttt{Livre}, \texttt{détester}(\texttt{marie}, \texttt{Dummy}))$

$z \in \texttt{Que}(\texttt{Livre}, \texttt{détester}(\texttt{marie}, \texttt{Dummy})) \Leftrightarrow z \in \texttt{Livre} \wedge \texttt{détester}(\texttt{marie}, z)$

$\texttt{c} \in \texttt{Livre} \wedge \texttt{dètester}(\texttt{marie}, \texttt{c})$

$\texttt{aimer}(\texttt{jean}, \texttt{c})$

$\exists z(z \in \texttt{Livre} \wedge \texttt{détester}(\texttt{marie}, z) \wedge \texttt{aimer}(\texttt{jean}, z))$.

The latter is the formula characterizing the Discourse Representation Structure

jean c marie
c ∈ Livre
détester(marie, c)
aimer(jean, c)

.

$\exists z(z \in \texttt{Livre} \wedge \texttt{détester}(\texttt{marie}, z) \wedge \texttt{aimer}(\texttt{jean}, z))$.

As the last example of this section, consider the sentence

Jean aime chaque livre que Marie déteste
JOHN LIKES EVERY BOOK WHICH MARY DETESTS
$(\nu)\,(\pi^r\, s\, o^\ell)\,(n\; c^\ell)(c)\,(c^r\; c\; \hat{o}\, \hat{s}^\ell)\;\; (\nu)\;\; (\pi^r\, \hat{s}\;\; \hat{o}^r)\;\;\; s^r$.

The links define the translation

$\texttt{aimer}(\texttt{jean}, \texttt{chaque}(\texttt{Que}(\texttt{Livre}, \texttt{détester}(\texttt{marie}, \texttt{Dummy}))))$

together with the instances of the non-logical axioms

$\texttt{chaque}(Q) = Q$

$z \in \texttt{Livre} \wedge \texttt{détester}(\texttt{marie}, z) \Leftrightarrow z \in Q$

$\texttt{aimer}(\texttt{jean}, Q) \wedge z \in Q \Rightarrow \texttt{aimer}(\texttt{jean}, z)$,

where Q abbreviates $\texttt{Que}(\texttt{Livre}, \texttt{détester}(\texttt{marie}, \texttt{Dummy}))$. From these, we derive the first order formula characterizing the Discourse Representation Structure corresponding to this sentence, namely

$\forall(z \in \texttt{Livre} \wedge \texttt{détester}(\texttt{marie}, z) \Rightarrow \texttt{aimer}(\texttt{jean}, z))$.

Note that the characterizing formula together with the non-logical axioms implies the translation of the sentence.

Intuitively, the constant Dummy represents an entity satisfying the relative clause. Hence the translation of a sentence with several occurrences of a relative pronoun introduces a new copy $\texttt{Dummy}_1, \texttt{Dummy}_2, \ldots$ for every new occurrence. The non-logical axiom schema (6) is in fact a separation schema for expressions.

We conclude this section by summing up the properties of the semantical interpretation

Translation:

A. each basic type of a lexical entry is translated by a functional or relational symbol,

B. each non-basic type of a lexical entry corresponds to an argument of a functional or relational symbol of the entry,

C. non-logical axioms are associated to some of the non-logical symbols,

D. in a string of types corresponding to a string of words, each occurrence of the constant `Dummy` is replaced by a new copy $\texttt{Dummy}_1$, $\texttt{Dummy}_2, \ldots$.

E. substitution, based on the links of the reduction, computes the translation of the sentence and the instances of the non-logical axioms.

These rules are formulated for the sample dictionaries above, which do not use double or higher adjoints. We call such a dictionary *standard*. Many dictionaries proposed up to now for (fragments of) modern European languages are not standard. In the next section, however, we shall see that every pregroup grammar is strongly equivalent to one with a standard dictionary.

4 Strong equivalence to standard dictionaries

We show in this section that every finite pregroup dictionary, hence every pregroup grammar in the sense of [Buszkowski], is strongly equivalent to a standard dictionary. Here *strongly equivalent* means that not only the dictionary has the same sentences, but also that the sentences have the same reductions. The restriction that the dictionary must be finite can be replaced by the weaker condition that the dictionary is of *finite complexity*. Complexity is a concept tailored to pregroups, generalizing order of types in AB-grammars. Moreover, if the original dictionary is finite, the strongly equivalent standard dictionary given by the proof can be effectively computed. This will be exploited in the next section when defining semantics for non-standard pregroup grammars. However, finiteness is only a sufficient, not a necessary condition for an effective interpretation.

Definition 2 (**Dictionaries, finite, standard, etc.**)

Let V be a non-empty set and B a partially ordered set. In the following we refer to V as *vocabulary* and to its elements as *words*. The free pregroup generated by B is denoted $P(B)$.

A *dictionary D over B for V* is a map from V to the set of subsets of $P(B)$. It is said to be *finite*, if the sets V, B and $D(v)$ are finite for every $v \in V$.

The set of *types* T_D of D is formed by the types belonging to some $D(v)$

$$T_D = \bigcup \{D(v) : v \in V\} \ .$$

A simple type t *occurs* in D if there is a string of simple types $X = s_1 ... s_n \in T_D$ such that $t = s_i$ for some i.

A dictionary D is *standard* if every simple type occurring in D is either basic or the right or the left adjoint of a basic type.

A dictionary D is of *rigidity* r, if every set $D(v)$ has at most r elements.

A *set* of types $T \subseteq P(B)$ has *width* w, if every string of simple types $s_1...s_k = X \in T$ has length $k \leq w$. A *dictionary* has width w if its set of types has this property.

A *type assignment* for a string $v_1...v_n$ of words is a string $X_1...X_n$ of types in $P(B)$ such that $X_i \in D(v_i)$, for $1 \leq i \leq n$. A *lexical entry* of D is a pair $v : X$ where $v \in V$ and $X \in D(v)$.

A sequence $v_1...v_n$ is an *s-sentence* of D if $X_1...X_n \to s$ for some type assignment $X_i \in D(v_i)$. The *s-language* of D is the set of its s-sentences. It is assumed that s is a basic type. If s is understood, we simply say *language* respectively *sentence* of D.

AB-grammars of order at most 1 provide examples of standard dictionaries. Indeed, the translation of $a_1\backslash...a_m\backslash b/c_1.../c_n$ into pregroups is

$$X = a_1^r...a_m^r b\, c_1^\ell...c_n^\ell,$$

where the a_i's c_j's and b are basic types. In a general standard grammar, however, more than one of the simple types may be basic. Moreover, basic types, right adjoints and left adjoints may intermingle arbitrarily.

Even standard pregroup dictionaries are more powerful than grammars based on Lambek Calculus. Recall the translation from Lambek Calculus to pregroup calculus: it maps $a \otimes b$ to ab, a/b to ab^ℓ and $b\backslash a$ to $b^r a$. This translation transforms a derivable sequent into a derivable inequality of types. However, the converse does not hold: A well-known example from Lambek Calculus is the non-derivable sequent $(a \otimes b)/c \not\vdash a \otimes (b/c)$. The left hand side and the right hand side translate to the same type, namely $(ab)c^\ell = a(bc^\ell)$. A fortiori, $(ab)c^\ell \to a(bc^\ell)$ holds in pregroups. Other examples can be found in [Moortgart-Oehrle].

Definition 3 (**Complexity**)
Let D be a pregroup dictionary and C a connected component of the set of basic types B. A sequence of simple types $s_0, \ldots, s_k$ occurring in D is *continuous* in C if there are a sequence of basic types $b_0 \in C, \ldots, b_k \in C$ and an integer z such that

$$s_0 = b_0^{(z)}, s_1 = b_1^{(z+1)}, \ldots,\ s_k = b_k^{(z+k)}\,.$$

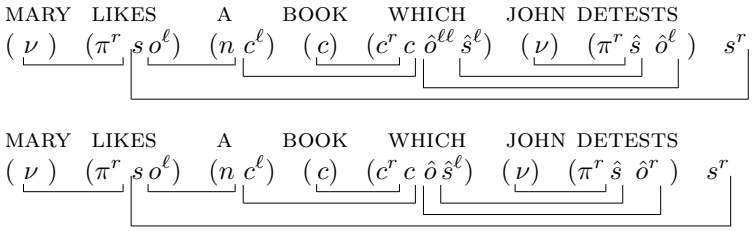
A dictionary D has *complexity* c if $k \leq c$ for every continuous sequence $s_0, \ldots, s_k$. A dictionary is of *finite complexity* if it has complexity c for some integer $c > 0$.

Complexity is the pregroup concept corresponding to order in Lambek Calculus. The former counts successive iterations of adjoints, the latter depth under the slashes / and \. In pregroups, however, counting does not necessarily start at basic types, it may start at different levels of iteration. Only the consecutive iterations are relevant. The presence of preorder makes it necessary to include

all elements of a connected component when counting consecutive iterators. For example, if only $a^{\ell\ell}$, a, a^{rrr} and a^{rrrrr} occur in D, then D has complexity 0. On the other hand, if all types in the dictionary are of the form $a_1^r...a_m^r b c_1^\ell...c_n^\ell$ it has complexity 2 in general, but if none of the a_i's or c_i's belongs to the connected components of the b's, it has complexity 0. A standard dictionary has complexity 2, but the converse is false. Note that a finite dictionary always is of finite width, rigidity or complexity.

Definition 4 (**Strong Equivalence**)
Dictionaries D and D' over the same vocabulary V are *strongly equivalent*, if for every type assignment $X_i \in D(v_i)$ and every reduction R that is a transition $R : X_1...X_n \Rightarrow s$ there exists a type assignment $X_i' \in D'(v_i)$ such that the strings X_i and X_i' have the same length and R is a transition from $X_1'...X_n'$ to s. The same must hold, if the roles of D and D' are exchanged.

For example, if we define the type of the relative pronoun WHICH as $c^r c\hat{o}^{\ell\ell}\hat{s}^\ell$ and for the definite verb-form DETESTS as $\pi^r \hat{s}\hat{o}^\ell$, the sentence below will have the same reduction in this grammar as in the standard grammar of the previous section:



The first step toward standard dictionaries consists in switching to a strongly equivalent centered dictionary.

Definition 5 (**Centered dictionaries**)
Let D be a pregroup dictionary and C a connected component of basic types. A sequence $b_0^{(z)}, b_1^{(z+1)}, \ldots, b_k^{(z+k)}$, continuous in C, is said to be *maximal* and the associated integer interval $I = \{z, z+1, ..., z+k\}$ is called a *complexity interval* of C if

for all $a \in C$ neither $a^{(z-1)}$ nor $a^{(z+k+1)}$ occur in D.

D is said to be *centered* if 0 belongs to every complexity interval.

All pregroup dictionaries designed so far for natural language processing are centered around 0. Also note that a connected component can have several mutually disjoint complexity intervals. If, however, the dictionary is centered then all complexity intervals of a connected component have a common element and are therefore identical.

Lemma 1 *Every dictionary of finite complexity can be replaced by a strongly equivalent centered dictionary of the same complexity, same width and same rigidity. Therefore the constant of the general parsing algorithm is the same for both dictionaries.*

Proof: Note that for every simple type $a^{(z)}$ occurring in D there are a unique connected component C and a unique complexity interval $I = \{v_I, \ldots, u_I\}$ such that $a \in C$ and $v_I \leq z \leq u_I$. The *midpoint* m_I of the complexity interval $I = \{v_I, v_I + 1, ..., u_I\}$ is defined by $m_I = v_I + d_I$ where d_I is the unique integer such that $2d_I - 1 \leq u_I - v_I \leq 2d_I$. Then $-d_I \leq z - m_I \leq d_I$ follows from $v_I \leq z \leq u_I$. For every connected component C and every complexity interval I of C we make a new copy

$C_I = \{a_I : a \in C\}$

and let B' be the disjoint union of these copies. Order $C_I \subseteq B'$ isomorphically to C, if m_I is even and by the dual order if m_I is odd. Then

$a^{(z)} \to b^{(z)}$ in $P(B)$ if and only if $a_I^{(z-m_I)} \to b_I^{(z-m_I)}$ in $P(B')$.

The dictionary D' is obtained from D, by replacing every simple type $a^{(z)}$ occurring in D by

$$\phi(a^{(z)}) = a_I^{(z-m_I)} .$$

It follows that for every reduction R

$R : s_1...s_m \Rightarrow 1$ if and only if $R : \phi(s_1)...\phi(s_m) \Rightarrow 1$.

Hence D and D' are strongly equivalent. Moreover, to see that D' is centered, note that a maximal continuous sequence $b_0^{(v_I)}, b_1^{(v_I+1)}, \ldots, b_k^{(u_I)}$ of D is mapped to a maximal continuous sequence of D' with corresponding complexity interval

$$I' = \{v'_I, v'_I + 1, ..., u'_I\}$$

where $v'_I = v_I - (v_I + d_I)$ and $u'_I = u_I - (v_I + d_I)$. It is then easily checked that the midpoint of I' is 0. Finally, the constant of the general parsing algorithm depends on the width and rigidity of the dictionary.

The next step, which constructs the standard dictionary associated to a centered dictionary, will increase rigidity. A careful look at the constant of the general algorithm shows that it remains the same for both dictionaries.

Lemma 2 (Escaping higher order) *For every dictionary D of complexity $c \geq 3$ there is a strongly equivalent dictionary D' of complexity $c-1$. Moreover, D' has the same width as D.*

Proof: By the preceding lemma, we may assume that D is centered. For every connected component C that has a necessarily unique complexity interval $I_C = \{v_C, \ldots u_C\}$ we decrease the upper limit $u_C \geq 2$ and increase the lower limit $v_C \leq -2$. This is achieved by adding new basic types to C.

Case I) (decreasing upper limits)

Define

$B' = B \cup \{\bar{a} : a \in C,\ C \text{ connected component of } B,\ u_C \geq 2\}$.

Order the new basic types by

$$\bar{a} \to \bar{b} \text{ if and only if } a \to b\,.$$

Suppose $a^{(z)}$ occurs in D and let C be the connected component of a. Define

$\phi(a^{(z)}) = a^{(z)}$, if $u_C < 2$ or if $z < u_C - 1$

and else

$$\phi(a^{(u_C)}) = \begin{cases} \bar{a}^{(1)} & \text{if } u_C \text{ is odd} \\ \bar{a}^{(0)} & \text{if } u_C \text{ is even} \end{cases} \qquad \phi(a^{(u_C-1)}) = \begin{cases} \bar{a}^{(0)} & \text{if } u_C \text{ is odd} \\ \bar{a}^{(-1)} & \text{if } u_C \text{ is even} \end{cases} .$$

We remark that

$$b^{(u_C-1)}a^{(u_C)} \to 1 \text{ if and only if } \phi(b^{(u_C-1)})\phi(a^{(u_C)}) \to 1.$$

The dictionary D' over B' is constructed from the entries in D by replacing every simple type t by $\phi(t)$ except that $t = a^{(u_C-1)}$ may remain unchanged.

$$D'(v) = \{s'_1 \dots s'_p : \exists s_1 \dots s_p \in D(v)\ \forall j(\ 1 \le j \le p \Rightarrow (s'_j = \phi(s_j) \vee\ \exists C\ \exists a \in C\ (s_j = a^{(u_C-1)} = s'_j))\}.$$

Consider a type assignment $X_i \in D(v_i)$, $1 \le i \le n$ and a reduction R such that $R : X_1 \dots X_n s^r \Rightarrow 1$. Write $X_1 ... X_n s^r$ as a string of simple types

$X_1 ... X_n s^r = s_1 ... s_m$.

Suppose $s_k = a^{(u_C)}$ where $a \in C$ and $u_C \ge 2$. Then s_k is linked to a simple type s_i on its left such that $s_i s_k \to 1$. Hence R contains the under-link

$$... \underline{s_i ... s_k} ...., \text{ where } i < k \text{ and } s_i s_k \to 1.$$

Therefore s_i has the form $s_i = b^{(u_C-1)}$ for some $b \in C$. Replace s_k and s_i by their respective ϕ-values, obtaining $s'_k = \phi(s_k)$ and $s'_i = \phi(s_i)$. By the remark above it follows that

$s'_i s'_k \to 1$.

The other simple types remain unchanged. In particular, if $s_j = b^{(u_C-1)}$ is not a left end-point, we have $s'_j = s_j$. Then $R : X'_1 ... X'_n s^r \Rightarrow 1$ and

$$s'_1 ... s'_m = X'_1 ... X'_n s^r$$

defines a type assignment $X'_i \in D'(v)$, $1 \le i \le n$.

Conversely, let $X'_i \in D'(v_i)$, $1 \le i \le n$, be a type assignment from D' and R' a reduction of $X'_1 \dots X'_n s^r = s'_1 \dots s'_m$ to the empty type 1. Consider an under-link $\{i, k\} \in R'$ with $i < k$. As $s'_i s'_k \to 1$, either both s'_i and s'_k belong to $P(B)$ or $s'_i = \bar{b}^{(z-1)}$ and $s'_k = \bar{a}^{(z)}$ where a and b belong both to one and the same connected component C of B such that $u_C \ge 2$. In the former case, we let $s_i = s'_i$ and $s_k = s'_k$. In the latter, we define $s_i = b^{(u_C-1)}$ and $s_k = a^{(u_C)}$. This defines a type assignment $X_i \in D(v_i)$, $1 \le i \le n$, such that $R' : X_1 \dots X_n s^r = s_1 \dots s_m \Rightarrow 1$. Hence D and D' are strongly equivalent.

Case II) (increasing the lower limits)

Define

$$\phi(a^{(v_C)}) = \begin{cases} \underline{a}^{(-1)} & \text{if } v_C \text{ is odd} \\ \underline{a}^{(0)} & \text{if } v_C \text{ is even} \end{cases} \qquad \phi(a^{(v_C+1)}) = \begin{cases} \underline{a}^{(0)} & \text{if } v_C \text{ is odd} \\ \underline{a}^{(1)} & \text{if } v_C \text{ is even} \end{cases},$$

whenever $v_C \le -2$ and continue the argument similarly to step I). After applying both steps, we have constructed a strongly equivalent centered dictionary of

complexity $c - 1$.

As every finite pregroup dictionary is of finite complexity, the following theorem is an immediate consequence.

Theorem 3 (Strong Equivalence) *A pregroup dictionary is strongly equivalent to a standard pregroup dictionary if and and only if it is of finite complexity. In particular every pregroup grammar in the sense of [Buszkowski] is strongly equivalent to a standard pregroup dictionary.*

Dictionaries with double or higher adjoints have a natural higher order interpretation. For example, suppose we assign the type $c^r c\hat{o}^{\ell\ell}\hat{s}^\ell$ to the relative pronoun *que* (WHICH). By the Grishin laws, we have the equality

$$c^r c\hat{o}^{\ell\ell}\hat{s}^\ell = c^r c(\hat{s}\hat{o}^\ell)^\ell \,.$$

This suggests to read the type as a two-argument function which transforms a noun and a verb-phrase to a noun-phrase. The construction of the equivalent standard dictionary replaces $\hat{o}^{\ell\ell}$ by $\hat{o}$. Therefore the higher order argument place $(\hat{s}\hat{o}^\ell)^\ell$ has become $\hat{o}\hat{s}^\ell$ in the strongly equivalent standard dictionary. But the latter has a natural first order interpretation. Hence compactness, the property which distinguishes compact bilinear logic from Lambek Calculus, makes pregroup dictionaries independent of an higher order interpretation. Technically, this is achieved by a derivable separation schema for formulas involving only symbols introduced by the translation.

5 The Comparative in English

So far we have used dummies to represent sets of individuals referred to implicitly when a relative pronoun is used. For similar reasons, dummies model ellipsis in comparative sentences. The dictionary presented below is not standard. However, the construction of the Equivalence Theorem in Section 4 transforms it into a strongly equivalent standard dictionary. This construction maps a simple type with an even iterator into a basic type. Moreover, it preserves left and right adjoints. Therefore the Translation Rules A. and B. of Section 3 may be reformulated thus

A. each simple type of the form $a^{(2z)}$ in a lexical entry is translated by a functional symbol,

B. each simple type of the form $b^{(2z+1)}$ in a lexical entry corresponds to an argument place of a functional symbol associated to a simple type in the same entry.

Consider the following sentences:

1. *Mary reads more novels than John*
2. *Mary reads more novels than poems*

3. *Mary reads more novels than John poems*

4. *Mary reads more novels than John writes poems*

The information an English speaker can extract from these sentences must be derivable from the interpretation. This is to say, the interpretation must imply

I. $\exists x \exists y(\forall z(z \in x \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{mary}, z)) \wedge \forall z(z \in y \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{john}, z)) \wedge \texttt{geq}(x, y))$

II. $\exists x \exists y(\forall z(z \in x \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{mary}, z)) \wedge \forall z(z \in y \Leftrightarrow z \in \texttt{Poem} \wedge \texttt{read}(\texttt{mary}, z)) \wedge \texttt{geq}(x, y))$

III. $\exists x \exists y(\forall z(z \in x \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{mary}, z)) \wedge \forall z(z \in y \Leftrightarrow z \in \texttt{Poem} \wedge \texttt{read}(\texttt{john}, z)) \wedge \texttt{geq}(x, y))$

IV. $\exists x \exists y(\forall z(z \in x \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{mary}, z)) \wedge \forall z(z \in y \Leftrightarrow z \in \texttt{Poem} \wedge \texttt{write}(\texttt{john}, z)) \wedge \texttt{geq}(x, y))$

respectively.[2] Here geq is a binary predicate, which is interpreted as a relation between sets, i.e. a *generalized quantifier*.

Comparing the formula with its corresponding sentence, we see that natural language is more succinct than logic. In the sentence, only the new constituent(s) of the mentally implied statement is (are) present. This new constituent is introduced by *than*, namely the new subject *John* in Sentence 1 and the new object *poems* in Sentence 2. In Sentence 3, both object and subject are different in the implied statement and are therefore explicit in the comparison. Finally, in Sentence 4, the words following *than* form a complete sentence. According to this analysis, the word *more* prepares the listener for a comparison of two quantities. The first quantity is in all four sample sentences the set of novels read by Mary. The second quantity, introduced by *than*, consists of the poems Mary reads, respectively the novels John reads and so on. Hence the natural language sentence determines two sets and compares them. The logic formula represents the comparison by the predicate symbol geq.

Quantities are either expressed by mass nouns or plural count nouns. We assume that the plural of a count noun is given with its singular in the dictionary. In English, count nouns sometimes stand for the whole, e.g. *Mary likes poems*, sometimes for a part, e.g. *Mary writes poems*. We will ignore this here and translate plurals by the same constant as the corresponding singular. Hence we add a new basic type p for plural count nouns and require that $p \to o$ and $p \to \pi$.

Syntax without semantics

The following simple dictionary for *more* and *than* recognizes the sentences above, but does not make sense in the semantics proposed here. Suppose we would have the entries

[2]Though x and y denote sets, the formulas remain first order because our predicate logic is two-sorted

more	: $p\tilde{p}^{\ell}p^{\ell}$	$\texttt{more}(x_2, x_1)$
than	: $\tilde{p}n^{\ell}$	$\texttt{than}(z)$
than	: $\tilde{p}p^{\ell}$	$\texttt{than}(z)$
novels	: p	Novel
poems	: p	Poem

The new type $\tilde{p}$ instead of p serves to prevent a reduction to the sentence type of **Mary reads more novels poems.* There are two entries for *than* so that both Sentence 1 and Sentence 2 can be recognized. In the case of Sentence 1, the reduction to the sentence type

Mary reads more novels than John
$(\nu)\ (\pi^r\ s\ o^{\ell})\ (p\ \tilde{p}\ p^{\ell})\ (p)\ (\tilde{p}\ n^{\ell})\ (\nu)\qquad s^r$

would give the translation

$\texttt{read}(\texttt{mary}, \texttt{more}(\texttt{Novel}, \texttt{than}(\texttt{John})))$.

Now the semantical problems start. How are we to interpret the functional symbol `more` ? Obviously, the claim that $\texttt{than}(\texttt{John})$ represents the quantity of novels read by John is unsustainable. Formula 1. is certainly not a logical consequence of this translation. What we need is a translation which is true if the quantity of novels read by Mary is greater than the quantity of novels read by John.

We have to resolve ourselves to invent more talkative types for *more* and *than*.

Syntax with semantics

Case of Sentence 1 : *Mary reads more novels than John.*

reads	: $\pi^r s o^{\ell}$	$\texttt{read}(x_1, x_2)$
more	: $p\, s^r \tilde{p}\, \tilde{p}^r\ \tilde{s}\, \pi^{\ell}\, p^{\ell\ell}\, s\, o^{\ell}\, \tilde{p}^{\ell} p^{\ell}$	$\texttt{Dummy}_1\ \texttt{More}(y_6, y_1)\ \texttt{geq}(y_2, y_5)$ $\texttt{id}(y_6)\ \ \texttt{rep}(y_3, y_4)$
novels	: p	Novel
than	: $\tilde{p}\, s^r\, p\, p^{\ell}$	$\texttt{Than}(z_1, z_2)\ \texttt{Dummy}_2$

The types for *more* and *than* are now explicit enough to recuperate the logical formulas. The type of *more* creates the implicit binary relational symbol `geq` comparing two quantities. Next, it introduces a functional symbol `More` and a constant $\texttt{Dummy}_1$ creating the quantity determined by the first statement. It also introduces a predicate symbol `rep` for the implicitly repeated verb. Finally, the unary functional symbol `id` repeats the object. The type of *than* introduces a set constant $\texttt{Dummy}_2$ and a binary functional symbol `Than` determining a quantity.

According to the Translation Rules A end B, the simple types with an odd iterator are numbered in the order of their occurrence in the entry from left to right and correspond to the argument places. Similarly, each simple type with an even iterator corresponds to a functional symbol according to the order given by the lexical entry. For example, the correspondences in the entries of *more* and *than* are as follows

more :

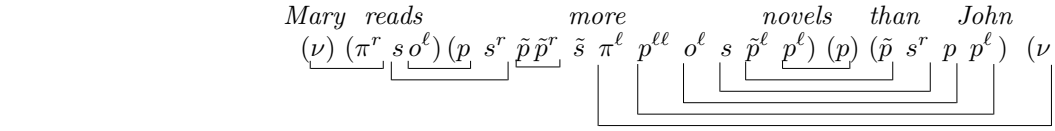
p	s^r	$\tilde{p}$	$\tilde{p}^r$	$\tilde{s}$	π^{ℓ}	$p^{\ell\ell}$	s	o^{ℓ}	$\tilde{p}^{\ell}$	p^{ℓ}
$\texttt{Dummy}_1$	y_1	More	y_2	geq	y_3	id	rep	y_4	y_5	y_6

than :

$\tilde{p}$	s^r	p	p^ℓ
Than	z_1	$\mathtt{Dummy}_2$	z_2

.

The reduction to the sentence type is



As the type of the sentence is rather long, we give the details of the procedure how to compute the translation and the relevant instances of the non-logical axioms from this reduction.

The links define the substitutions

$[x_1, x_2|\mathtt{mary}, \mathtt{Dummy}_1]$
$[y_1, y_2, y_3, y_4, y_5, y_6|\mathtt{read}, \mathtt{More}, \mathtt{john}, \mathtt{Dummy}_2, \mathtt{Than}, \mathtt{Novel}]$
$[z_1, z_2|\mathtt{rep}, \mathtt{id}]$

The expressions constructed by the appropriate substitutions are therefore

$\mathtt{read}(\mathtt{mary}, \mathtt{Dummy}_1)$
$\mathtt{More}(\mathtt{Novel}, \mathtt{read}(\mathtt{mary}, \mathtt{Dummy}_1))$
$\mathtt{rep}(\mathtt{john}, \mathtt{Dummy}_2))$
$\mathtt{Than}(\mathtt{id}(\mathtt{Novel}), \mathtt{rep}(\mathtt{john}, \mathtt{Dummy}_2))$
$\mathtt{geq}(\mathtt{More}(\mathtt{Novel}, \mathtt{read}(\mathtt{mary}, \mathtt{Dummy}_1)),$
$\mathtt{Than}(\mathtt{id}(\mathtt{Novel}), \mathtt{rep}(\mathtt{John}, \mathtt{Dummy}_2))).$

The truth of the whole sentence depends on the value of geq for two quantities, namely those substituted in places y_2 and y_5. The first quantity is defined by More, the second by Than. Both operators act like the restrictive relative pronoun in Section 3. Hence the

Non-logical axioms

(7.1) $z \in \mathtt{More}(X, p(\mathtt{Dummy}_1)) \Leftrightarrow z \in X \wedge p(z)$
(7.2) $z \in \mathtt{Than}(X, p(\mathtt{Dummy}_2)) \Leftrightarrow z \in X \wedge p(z)$,

where X is a set expression, p a predicate expression and z is accessible in p,

(7.3) $\mathtt{rep} = y_1$.

The latter is not a formula, but a shorthand how to construct the non-logical axiom. After substitution of y_1 by the predicate symbol given by the reduction, for example read, it becomes an equality of two functional symbols

$\mathtt{rep} = \mathtt{read}$,

which stands for the first order formula

$\forall x \forall y(\mathtt{rep}(x, y) = \mathtt{read}(x, y))$.

Applying the substitution to the non-logical axioms, we obtain the instances

$\mathtt{rep} = \mathtt{read} \quad \mathtt{id}(\mathtt{Novel}) = \mathtt{Novel}$

Therefore the translation is equal to

$\mathtt{geq}(\mathtt{More}(\mathtt{Novel}, \mathtt{read}(\mathtt{mary}, \mathtt{Dummy}_1)),$
$\mathtt{Than}(\mathtt{Novel}, \mathtt{read}(\mathtt{John}, \mathtt{Dummy}_2)))$.

The relevant instances concerning More and Than are

$z \in \texttt{More}(\texttt{Novel}, \texttt{read}(\texttt{mary}, \texttt{Dummy}_1)) \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{mary}, z)$
$z \in \texttt{Than}(\texttt{Novel}, \texttt{read}(\texttt{john}, \texttt{Dummy}_2)) \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{john}, z)$.

The logical content of the translation can now easily be derived, i.e.

$\exists x \exists y(\forall z(z \in x \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{mary}, z)) \wedge \forall z(z \in y \Leftrightarrow z \in \texttt{Novel} \wedge \texttt{read}(\texttt{john}, z)) \wedge \texttt{geq}(x, y))$.

Case of Sentence 2 : *Mary reads more novels than poems.*

The analysis of Sentence 2 is similar to that of Sentence 1. This time the verb and the subject are repeated, but the two objects are different. We add a new entry for *more* to our dictionary, whereas the other translations remain unchanged.

$$\begin{array}{lll} \textit{more} & : \; p\, s^r \pi^{rr} \pi^r \tilde{p}\, \tilde{p}^r \tilde{s}\, s\, o^\ell \tilde{p}^\ell p^\ell & \texttt{Dummy}_1\ \texttt{id}(y_2)\ \texttt{More}(y_6, y_1) \\ & & \texttt{geq}(y_3, y_5)\ \texttt{rep}(y_2, y_4) \end{array} .$$

In this case, the functional symbol `id` is associated to the basic type π^{rr} and depends on y_2. Therefore it repeats the common subject of the explicit and the implicit statement. The non-logical axioms (7.1)-(7.3) remain unchanged. Formula II. can be derived as in the previous example.

Case of Sentence 3 : *Mary reads more novels than John poems.*

Neither subject nor object are implicitly repeated. Hence, the type of *more* only repeats the verb by introducing `rep` associated to the basic type s as in the other cases. The dictionary entries are

$$\begin{array}{lll} \textit{more} & p\, s^r \tilde{p}\, \tilde{p}^r\, \tilde{s}\, \pi^\ell\, o^\ell\, s\, \tilde{p}^\ell p^\ell & \texttt{Dummy}_1\ \texttt{More}(y_6, y_1)\ \texttt{geq}(y_2, y_5) \\ & & \qquad\qquad\qquad\qquad\qquad\qquad \texttt{rep}(y_3, y_4) \\ \textit{than} & \tilde{p}\, s^r\, p\, \pi\, p^\ell \pi^\ell & \texttt{Than}(z_1, z_2)\ \texttt{Dummy}_2\ \texttt{id}(z_3) \end{array}$$

The simple types π and π^ℓ figure in the type of *than* for purely syntactical reasons. Their joint effect is to change the order of the two words *John poems* to *poems John*. This is achieved by the functional symbol $\texttt{id}(z_3)$. Again the non-logical axioms (7.1)-(7.3) remain unchanged.

The reduction to the sentence type, the interpretation of the sentence and the characterizing Formula III. are derived as in the other cases.

Case of Sentence 4 : *Mary reads more novels than John writes poems.*

The statement defining the second quantity is explicit. Hence the type of *more* no longer needs types for implicit repetition. Only the implicitly understood comparing relation `geq` is still present besides `More` and $\texttt{Dummy}_1$.

$$\begin{array}{lll} \textit{more}\colon & p\, s^r \tilde{p}\, \tilde{p}^r\, \tilde{s}\, \tilde{p}^\ell p^\ell & \texttt{Dummy}_1\ \texttt{More}(x_4, x_1)\ \texttt{geq}(x_2, x_3) \\ \textit{than}\ : & \tilde{p}\, p^\ell\, p^{\ell\ell}\ s^\ell & \texttt{Than}(y_1, y_2)\ \texttt{Dummy}_2 \end{array}$$

The non-logical axioms are those concerning `More` and `Than`, namely (7.1) and (7.2). Formula IV. can now be derived as in the other cases. We remark that in spite of the four different entries for *more* and the three different ones for *than*, the meaning of `More` and `Than` as reflected by the non-logical axioms remains unchanged.

The relatively long type associated to a single word like *more* is the price to pay for the substitution algorithm which computes the translation of a sentence

for every transition to the sentence type. As the algorithm is linear, the extra effort required for a dictionary with semantics may be worth while paying.

6 Conclusion

Our approach is similar in spirit, but not in form, to that given in [Fenstad et al.]. Dictionaries list each syntactical entry with a semantical translation. The insistence that the translation of the sentence must be pieced together from the translations of the individual words makes interpretation a function defined for reductions of sentences. The translation can be computed by an algorithm which runs in time proportional to the size of the reduction.

Moreover, the strong equivalence theorem shows that reduction, i.e. the geometrical structure of pregroup derivations, is the invariant which caries the semantical meaning. The strong equivalence of 'higher order' with 'first order' pregroup grammars is the syntactical counterpart for equivalence of fragments of higher order logic with first order logic. The only functional symbols depending on predicate symbols introduced so far are `Que (Which)`, `More` and `Than`. They are definable by comprehension restricted to expressions from the natural language. Axiom-schema (1) is the corner stone on which the interpretation rests. The others concern particular words, which are interpreted as quantifiers, comprehension operators etc. The working assumption here is that the logical and set theoretical content of words can be expressed within the persistent fragment of second order logic. This fragment is effectively axiomatizable and equivalent to a two-sorted first order logic with a primitive $\in$. There, comprehension axioms for set existence can be expressed as first order axioms. A more comprehensive language fragment will have to be analysed with pregroup grammars to confirm that the corresponding semantics stays in the frame outlined in [van Benthem].

The strong equivalence theorem assures that nothing is lost in expressiveness by restricting attention to standard grammars. A semantically meaningful pregroup grammar may require more, or more complicated types per word than one intended for mere syntactical recognition. However, instead of making the design of a pregroup grammar more difficult, semantics may make it more easy: once we know the semantic translation of the words, the corresponding types can be learned, see [Béchet-Forêt-Tellier]. Moreover, the proliferation of types does not necessarily increase the complexity of parsing. In fact, the dictionaries proposed here can be parsed by an algorithm with a runtime proportional to the length of the sentence.

The step toward an interpretation in categorical logic is quite feasible, if based on a category satisfying functional completeness in the sense of [Lambek-Scott]. Indeed, a functionally complete category simulates substitution of terms as composition of arrows. An example how to compute an interpretation using composition in a *2*-category is given in [Preller]. Interpretation in a categorical system incorporating higher order logic will also permit a comparison between pregroup grammars and Head-driven Phrase Structure Grammars,

following the higher order categorical grammars of [Pollard].

For many reasons, the ideas proposed in the present paper can only be a first step toward semantics for pregroup grammars. In a next step, the tree-structure of the translations could be exploited to describe the accessibility conditions of discourse referents and develop an anaphoric theory by adding equalities as non-logical axioms.

7 Acknowledgments

I would like to thank Carl Pollard for having asked the question about the linguistic significance of compactness, Michele Abrusci for having rephrased the strong equivalence theorem as independence of discourse from higher order logic and Joachim Lambek for having added compactness to his Syntactic Calculus.

References

[Béchet-Forêt-Tellier] Denis Béchet, Annie Forêt, Isabelle Tellier, Learnability of Pregroup Grammars, International Conference on Grammatical Inference, Springer, LNAI 3262, pp. 65 - 76, 2004

[Buszkowski] Wojciech Buszkowski, Lambek Grammars based on pregroups, in: P. de Groote et al., editors, Logical Aspects of Computational Linguistics, LNAI 2099, Springer, 2001

[Degeilh-Preller] Sylvain Degeilh, Anne Preller, Efficiency of Pregroups and the French noun phrase, Journal of Language, Logic and Information, Springer, Vol. 14, Number 4, pp. 423-444, 2005

[Earley] Jay Earley, An efficient context-free parsing algorithm, Communications of the AMC, Volume 13, Number 2, pp 94-102, 1970

[Fenstad et al.] Jens Erik Fenstad, Per-Kristian Halvorsen, Tore Langholm, Johan van Benthem, Situations, Language and Logic, *Studies in Linguistics and Philisophy*, Reidel, 1987

[Kamp-Reyle] Hans Kamp, Uwe Reyle, From Discourse to Logic, Introduction to Modeltheoretic Semantics of Natural Language, Kluwer Academic Publishers, 1993

[Lambek 1958] Joachim Lambek, The mathematics of sentence structure, American Mathematical Monthly 65, pp.154-170, 1958

[Lambek-Scott] Joachim Lambek, Philipp Scott, Introduction to higher order categorical logic, Cambridge University Press, 1986

[Lambek 1999] Joachim Lambek, Type Grammar revisited, in: A. Lecomte et al., editors, Logical Aspects of Computational Linguistics, Springer LNAI 1582, pp.1 -27, 1999

[Lambek 2004] Joachim Lambek, A computational algebraic approach to English grammar, Syntax 7:2, pp. 128-147, 2004

[Moortgart-Oehrle] Michael Moortgat, Richard Oehrle, Pregroups and Type-Logical Grammar: Searching for Convergence, Language and Grammar Studies in Mathematical Linguistics and Natural Language, P. Scott, C. Casadio, R.A.Seeley (eds.), CSLI Publications, 2004

[Pollard] Carl Pollard, Higher-Order Categorical Grammars, Proceedings of Categorial Grammars 04, Montpellier, France, 2004

[Preller-Lambek] Anne Preller, Joachim Lambek, Free compact 2-categories, Mathematical Structures for Computer Sciences, Cambridge University Press, accepted October 2005

[Preller] Anne Preller, Category Theoretical Semantics for Pregroup Grammars, Philippe Blache and Edward Stabler (Eds.): LACL 2005, LNAI 3492, pp. 254-270, Springer 2005

[van Benthem] Johan van Benthem, Guards, Bounds and Generalized Semantics, Journal of Language, Logic and Information, Springer, Vol. 14, pp. 263-279, 2005