

Computing Italian clitic and relative clauses with tupled pregroups

Claudia Casadio
Aleksandra Kiślak-Malinowska

◇Dept. of Philosophy, Univ. G. D’Annunzio, Chieti IT - casadio@unich.it
‡Faculty of Mathematics and Computer Science
Univ. of Warmia and Mazury, Olsztyn PL - akis@uwm.edu.pl

Abstract. Pregroup grammars are introduced by Lambek in 1999 as an algebraic tool for the formal analysis natural languages. The main focus of the present paper is placed on a special extension of this calculus known as tupled pregroup grammars. Their applications to Italian concerning different sentential structures involving clitic pronouns, relative pronouns and related phenomena are taken into consideration and the advantages of the tupled pregroup approach are shown from the point of view of both linguistic analysis and syntactic computation.

KEYWORDS: Pregroups, tupled pregroup grammars, natural language, clitic pronouns, relative clauses

1 Introduction

Some years ago E. Stabler [33] developed the idea of extending pregroups into tupled pregroup grammars and showed how to make use of them to treat some crucial aspects of English syntax. According to him the pregroup operations provide a simple feature checking and the tupling allows additional operations like ‘movement’. Languages definable in tupled pregroup grammars are weakly equivalent to multiple context-free grammars with the power to define the mildly context sensitive languages like $\{xx|x \in \{a,b\}^*\}$, $\{a^n b^n c^n | n \geq 1, a, b, c \in \Sigma\}$, $\{a^m b^n c^m d^n | m, n \geq 1, a, b, c, d \in \Sigma\}$.

In this paper ¹, following Stabler’s proposal, we introduce a tupled pregroup grammar for a fragment of Italian grammar including clitics, relative pronouns and related phenomena. The result obtained has the advantage of involving local applications of the rules and in the same time is consistent with the *move* vs. *merge* strategy put forward by Stabler in coherence with the Chomskian minimalist theory of grammar.

¹ The authors are grateful to Jim Lambek and Wojciech Buszkowski for stimulating suggestions in the course of a number of discussions during the past years.

Thanks are also due to the anonymous referees for useful comments. Support by MIUR (PRIN 2012, 60% 2013) is acknowledged by C. Casadio.

2 Tupled pregroup grammars

The calculus of pregroups is developed by Lambek [20, 21] as an alternative to his Syntactic Calculus [19], a well known model of categorial grammar [24, 25]. Pregroups are a particular kind of substructural logic that is compact and non-commutative [6, 7]. In fact the calculus is a non-conservative extension of classical non-commutative multiplicative linear logic [1, 10]: the *left* and *right* ‘iterable’ adjoints of pregroups have as their counterparts the *left* and *right* ‘iterable’ negations of non-commutative multiplicative linear logic; in principle formulas can occur with n (left or right) adjoints ($n \geq 2$), although 2 appears as being sufficient for linguistic applications; see [10, 21, 16].

In summary, a *pregroup* $\{G, \cdot, 1, {}^\ell, {}^r, \rightarrow\}$ is a partially ordered monoid in which each element a has a *left adjoint* a^ℓ , and a *right adjoint* a^r such that

$$\begin{aligned} a^\ell a &\rightarrow 1 \rightarrow a a^\ell \\ a a^r &\rightarrow 1 \rightarrow a^r a \end{aligned}$$

where the dot “ $\cdot$ ”, that is usually omitted, stands for the pregroup operation, the compact conjunction or multiplication with unit 1, the arrow denotes the partial order, the rules $a^\ell a \rightarrow 1$, $a a^r \rightarrow 1$ are called *contractions*, and the opposite rules $1 \rightarrow a a^\ell$, $1 \rightarrow a^r a$ are called *expansions*. From the point of view of linguistic analysis, the constant 1 represents the empty string of types, and the operation “ $\cdot$ ” is interpreted as concatenation.

Contractions like $a^\ell a \rightarrow 1$ and $a a^r \rightarrow 1$ are the basic means to determine if a string of words, to which pregroups types are assigned, is grammatical and, particularly, if it is a sentence. A pregroup grammar is freely generated by a partially ordered set of *basic* types $a_1, \dots, a_n$: from each basic type a we form *simple* types by taking single or repeated adjoints: $\dots a^{\ell\ell}, a^\ell, a, a^r, a^{rr} \dots$. A compound type or just a *type* is a string of simple types. For example, assigning the basic type n to the syntactic category of noun phrases, and the simple type $(n^r s)$ to the syntactic category of verb phrases, one obtains that the concatenation (product) of type n with type $n^r s$ leads to the type s of a sentence by applying the contraction $(n \cdot (n^r s)) \rightarrow s$.

The types assigned to the words of a given language are assumed to be stored permanently in the speaker’s ‘mental’ dictionary, and *metarules* are introduced into the dictionary to simplify lexical assignments and make syntactic calculations quicker [20, 21]. Eventually metarules can be used to introduce semantic information, depending on the language under consideration; see e.g. [3]. In recent years several works have been developed to account for the semantics of pregroups and interesting connections have been individuated with compact monoidal categories, within category theoretical semantics [22, 28]. This field appears particularly promising for its relations to research in quantum logic and computation, with particular reference to vector space semantics. We cannot enter in details here, but we refer in particular to [27, 29].

Given their ‘simple’ type logical nature and their computational efficiency, pregroups have been applied, in a short span of time, to a wide range of languages from English and French to Polish, Arabic and Persian; see e.g. [9, 2, 3, 11, 16,

32]. Also their computational properties have been studied in several directions, e.g. [15, 4, 5, 14].

In this paper we will present a new approach to pregroups known as tupled pregroup grammar. In tupled pregroup grammars the lexicon consists of tuples of ordered pairs whose first element is a type and the second one is a symbol from the alphabet. Elements of the lexicon are called *lexical entries*. The idea of taking certain elements of an alphabet together in one tuple can be motivated and explained with the fact that in natural languages certain words tend to occur together in the sentence, as for example *prepositions* with *nouns*, *pronouns* with *verbs*, *clitic pronouns* with certain *verbs*, etc. In formal languages one may wish to copy each item of an alphabet (as in copying language $\{xx|x \in \{a, b\}^*\}$), to have the same number of occurrences of certain elements, etc.

Let s_i be elements of a finite alphabet $\Sigma \cup \epsilon$ (we can think of them as the words of a natural language, ϵ being an empty string), and let $\mathbb{P}$ be a set of simple types, partially ordered by $\leq$. Types $(\mathbb{T}_{\mathbb{P}})$ are of the form $p_1^r p_2^r \dots p_k^r v w_1^l w_2^l \dots w_m^l$, for $p_1, p_2, \dots, p_k, v, w_1, w_2, \dots, w_m \in \mathbb{P}$, $k, m \geq 0$, where p^r and p^l are called *right* and *left* adjoints of p , respectively, for any $p \in \mathbb{P}^2$. The latter fulfil the following inequations³: $p^l p \leq 1$ and $pp^r \leq 1$.

The lexical entries take the following form:

$$\begin{pmatrix} t_1 & t_2 & \dots & t_k \\ s_1 & s_2 & \dots & s_k \end{pmatrix}$$

Here $s_1, \dots, s_k$ are elements of the alphabet and $t_1, \dots, t_k$ are types.

A *merge operation* applying to any pair of tuples is defined as follows:

$$\begin{pmatrix} t_1 & \dots & t_i \\ s_1 & \dots & s_i \end{pmatrix} \bullet \begin{pmatrix} t_{i+1} & \dots & t_k \\ s_{i+1} & \dots & s_k \end{pmatrix} = \begin{pmatrix} t_1 & \dots & t_k \\ s_1 & \dots & s_k \end{pmatrix}$$

An *operation of deleting i-th coordinate*, for any k-tuple $k > 0$ and any $1 \leq i \leq k$ is defined as follows:

$$\begin{pmatrix} t_1 & \dots & t_{i-1} & t_i & t_{i+1} & \dots & t_k \\ s_1 & \dots & s_{i-1} & s_i & s_{i+1} & \dots & s_k \end{pmatrix}_{-i} = \begin{pmatrix} t_1 & \dots & t_{i-1} & t_{i+1} & \dots & t_k \\ s_1 & \dots & s_{i-1} & s_{i+1} & \dots & s_k \end{pmatrix}$$

Let us define a binary relation on tupled pregroup expressions, denoted by $\Rightarrow$ that holds in the following cases, for any tuples e_1, e_2 and sequence of tuples α, β :

$$\begin{aligned} (Mrg) \quad & \alpha \ e_1 \ e_2 \ \beta \Rightarrow \ \alpha \ e_1 \bullet e_2 \ \beta \\ (Move) \quad & \alpha \ \begin{pmatrix} t_1 & \dots & t_k \\ s_1 & \dots & s_k \end{pmatrix} \beta \Rightarrow \alpha \ \begin{pmatrix} t_i t_j \\ s_i s_j \end{pmatrix} \bullet \begin{pmatrix} t_1 & \dots & t_k \\ s_1 & \dots & s_k \end{pmatrix}_{-i-j} \beta \end{aligned}$$

² Every type must contain exactly one basic type v (without the superscript r or l) and may contain an arbitrary number (possibly equal to zero) of right and left adjoints.

³ Presenting types in that form is a great simplification, but it is done on purpose for our linguistic applications; for more details see [33].

(*Move*) applies to any k -tuple ($k > 1$), for any $1 \leq i \leq k$ and $1 \leq j \leq k$.

The type in any coordinate can be contracted, for any a, b such that $a \leq b$:

$$\begin{aligned} (GCon) \quad & \alpha \left(\begin{array}{ccc} x & b^l a y & \\ & s & \end{array} \right) \beta \Rightarrow \alpha \left(\begin{array}{ccc} xy & & \\ & s & \end{array} \right) \beta \\ (GCon) \quad & \alpha \left(\begin{array}{ccc} x & a b^r y & \\ & s & \end{array} \right) \beta \Rightarrow \alpha \left(\begin{array}{ccc} xy & & \\ & s & \end{array} \right) \beta \end{aligned}$$

Additionally, (*Mrg*) can be applied to a pair of tuples only when in one tuple all types are of the form v (without left and right adjoints) and (*Move*) takes two items of a tuple if one of the types is of the form v .

A tupled pregroup grammar is $G = (\Sigma, \mathbb{P}, \leq, \mathbb{I}, S)$, where Σ is a finite alphabet, $\mathbb{P}$ is a set partially ordered by $\leq$, $\mathbb{I}$ is a relation $\mathbb{I} \subset (\mathbb{T}_{\mathbb{P}} \times (\Sigma \cup \{\epsilon\})^*)$, S is a designated type (here the type of a sentence). The language of the grammar is

$$L_G = \{s | \mathbb{I}^* \xRightarrow{*} \binom{S}{s}\}$$

3 Applying tupled pregroup grammar to chosen aspects of Italian grammar

We now introduce an application of a tupled pregroup grammar to a well known aspect of the Italian language: clitic pronouns placement and occurrence within certain relative clauses⁴. Single tuples are put together with those containing two or three coordinates - the first word in the tuple being usually a verb in its inflected form (remind again that the order of pairs (type, word) within a tuple is not important and it can be switched without any consequences). Empty strings play the role of ordering the words or strings of words within the sentence.

The idea behind the tuples definitions is that some words are indispensable in sentences with clitic pronouns, and have to occur together in order to fulfill certain features and peculiarities of the grammar. In comparison with former approaches (see [9],[11]) the types assigned to the words are less complicated and not so numerous; nonetheless, the proposed tupled pregroup grammar ‘catches’ all the acceptable clitic patterns and excludes those considered wrong by Italian grammarians (see e.g. [23]).

Italian clitics exhibit two basic patterns: clitic pronouns can occur both in pre-verbal and post-verbal position, keeping the same relative order of occurrence: locative/indirect object/direct object. Subjects do not allow clitic counterparts and concerning the other verbal arguments, clitic unstressed elements can be attached both to the main verb or to the auxiliaries and the modal verbs. Therefore the set of clitic types will include types for the *direct* object

⁴ On the subject of clitic pronouns in Romance languages, with particular reference to French and Italian, there is an extensive and articulated literature, both in the field of theoretical linguistics and of type logical grammars. We cannot enter here into details, but cf. [31, 23, 26]; on the side of pregroups, we refer to the two leading papers: [2, 9].

(accusative), types for the *indirect* object (dative), and types for the *locative* object. The types used in our examples are:

- s_1 type of a sentence in a present tense
- π_3 third person subject
 - we assume $\pi_3 \leq o_{m4} \leq o_4$
 - (The assumption may be interpreted this way: third person subject in the sentence can also play the role of a direct object with masculine gender and the latter may be treated as a direct object when gender does not matter.)
- n_m noun masculine singular
- n_f noun feminine singular
- N_m noun phrase masculine singular
 - for example *un gatto, il cane*
 - we assume that $N_m \leq \pi_3$ and $N_m \leq o_{4m} \leq o_4$
- N_f noun phrase feminine singular
 - for example *una foto, la foto*
 - we assume that $N_f \leq \pi_3$ and $N_f \leq o_{4f} \leq o_4$
- i infinitive of the verb, for example *dare*
- $\bar{i}$ infinitive form of the verb phrase with a clitic,
 - for example *darla, darmi, darmela, poterla avere vista*
- o_4 an accusative (direct) object (as i.e. *una ragazza, una bella ragazza*, etc.)
- o_3 a dative (indirect) object (as *a me, a una ragazza*, etc.)
- $\hat{o}_4$ an accusative (direct) personal pronoun occurring within a tuple,
 - accompanied with the verb as in $\begin{pmatrix} i & \hat{o}_4 \\ dare & la \end{pmatrix}$
- $\hat{o}_3$ a dative(direct) personal pronoun occurring within a tuple,
 - accompanied with the verb as in $\begin{pmatrix} i & \hat{o}_3 \\ dare & mi \end{pmatrix}$
- $\check{o}_3$ a dative (direct) personal pronoun occurring within a tuple, accompanied
 - with the verb and an accusative personal pronoun as in $\begin{pmatrix} i & \check{o}_3 & \hat{o}_4 \\ dare & me & la \end{pmatrix}$
- Note: we do not compare either o_4 with $\hat{o}_4$ or o_3 with $\check{o}_3$.
- p_2 the plain form of past participle (*visto, dato*) of the verb
 - which takes an auxiliary verb *avere* to form past participle
- $\bar{p}_2$ the full form of past participle (*visto una ragazza, dato una mela a me*)
 - of the verb with an auxiliary verb *avere*
 - and we assume that $p_2 \leq \bar{p}_2$

Let the dictionary (showing only tuples used in our examples) be as follows, for more see [12]:

$$I = \left\{ \begin{pmatrix} \bar{i}\bar{p}_2^l \\ avere \end{pmatrix} \begin{pmatrix} p_2 & \check{o}_3 & \hat{o}_4 \\ data & me & la \end{pmatrix} \begin{pmatrix} s_1\bar{p}_2^l \\ ha \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l \\ ha \end{pmatrix} \begin{pmatrix} \pi_3 \\ Mario \end{pmatrix} \begin{pmatrix} i & \hat{o}_4 \\ vedere & la \end{pmatrix} \right. \\
\begin{pmatrix} \bar{p}_2 \bar{i}^l \\ voluto \end{pmatrix} \begin{pmatrix} s_1 \bar{i}^l \\ vuole \end{pmatrix} \begin{pmatrix} N_m n_m^l \\ il \end{pmatrix} \begin{pmatrix} N_m n_m^l \\ un \end{pmatrix} \begin{pmatrix} N_f n_f^l \\ una \end{pmatrix} \begin{pmatrix} n_m \\ gatto \end{pmatrix} \begin{pmatrix} n_f \\ foto \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 \\ dorme \end{pmatrix} \\
\begin{pmatrix} \pi_3^r s_1 o_4^l \\ vede \end{pmatrix} \begin{pmatrix} i & \hat{o}_4 \\ guardare & lo \end{pmatrix} \begin{pmatrix} w_1 & n_m \\ che & \epsilon \end{pmatrix} \begin{pmatrix} w_2 & x \\ cui & \epsilon \end{pmatrix} \begin{pmatrix} s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \epsilon \end{pmatrix} \begin{pmatrix} s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 \\ \epsilon \end{pmatrix} \\
\left. \begin{pmatrix} w_1^r \pi_3^r \pi_3 s_1^l \\ \epsilon \end{pmatrix} \begin{pmatrix} w_2^r n_m n_m^l \\ \epsilon \end{pmatrix} \begin{pmatrix} x^r \pi_3^r \pi_3 s_1^l \\ \epsilon \end{pmatrix} \dots \right\}$$

Example 1 Let us start with a very simple example **Mario vede Dario** (*[Mario] [sees] [Dario] - Mario sees Dario*) with the following lexical entries

$$\begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} N_m \\ Dario \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l \\ vede \end{pmatrix}$$

Using the rules of (*Mrg*), (*Move*) and (*GCon*) we can justify the correctness of this sentence with the following derivation:

$$\begin{aligned} & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l \\ vede \end{pmatrix} \begin{pmatrix} N_m \\ Dario \end{pmatrix} \Rightarrow (\text{Mrg on the second and third tuples}) \\ & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l & N_m \\ vede & Dario \end{pmatrix} \Rightarrow N_m \leq o_4 \text{ (making use of partial order)} \\ & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l & o_4 \\ vede & Dario \end{pmatrix} \Rightarrow (\text{Move on coordinates of the second tuple}) \\ & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l o_4 \\ vede & Dario \end{pmatrix} \Rightarrow (\text{GCon in the second tuple}) \\ & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 \\ vede & Dario \end{pmatrix} \Rightarrow (\text{Mrg on the first and the second tuple}) \\ & \begin{pmatrix} N_m & \pi_3^r s_1 \\ Mario & vede & Dario \end{pmatrix} \Rightarrow N_m \leq \pi_3 \text{ (making use of partial order)} \\ & \begin{pmatrix} \pi_3 & \pi_3^r s_1 \\ Mario & vede & Dario \end{pmatrix} \Rightarrow (\text{Move on coordinates of the tuple}) \\ & \begin{pmatrix} \pi_3 \pi_3^r s_1 \\ Mario & vede & Dario \end{pmatrix} \Rightarrow (\text{GCon within the tuple}) \\ & \begin{pmatrix} s_1 \\ Mario & vede & Dario \end{pmatrix} \end{aligned}$$

Sometimes the derivation is presented in graphical form by means of non-crossing underlinks:

$$\begin{array}{ccc} Mario & vede & Dario \\ N_m & \pi_3^r s_1 o_4^l & N_m \end{array} \Rightarrow s_1 \text{ (here } N_m \leq o_4 \text{ and } N_m \leq \pi_3)$$

The above given example can be done in classical pregroup grammars - the example serves only to see how the mechanism of derivation of the sentence works. The examples with tuples with more coordinates will be shown below.

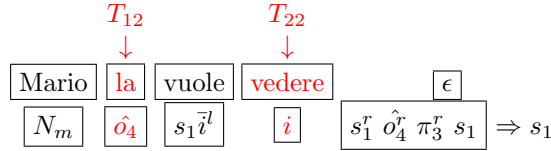
Example 2

Using the rules of (*Mrg*), (*Move*) and (*GCon*) we can justify the correctness of the sentence **Mario la vuole vedere** (*[Mario] [her] [wants] [(to) see] - Mario wants to see her*) and proceed as follows

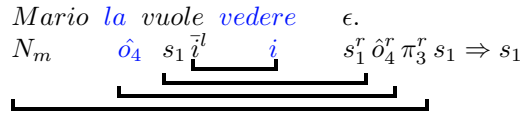
$$\begin{aligned} & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} s_1 \bar{i}^l \\ vuole \end{pmatrix} \begin{pmatrix} i & \hat{o}_4 \\ vedere & la \end{pmatrix} \begin{pmatrix} s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \epsilon \end{pmatrix} \Rightarrow \\ & \begin{pmatrix} N_m \\ Mario \end{pmatrix} \begin{pmatrix} s_1 \bar{i}^l & i & \hat{o}_4 \\ vuole & vedere & la \end{pmatrix} \begin{pmatrix} s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \epsilon \end{pmatrix} \Rightarrow \end{aligned}$$

$$\begin{aligned}
& \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{cc} s_1 \bar{i}^l i & \hat{o}_4 \\ \text{vuole vedere la} & \end{array} \right) \left(\begin{array}{c} s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \epsilon \end{array} \right) \Rightarrow \textcolor{yellow}{i \leq \bar{i}} \\
& \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{cc} s_1 & \hat{o}_4 \\ \text{vuole vedere la} & \end{array} \right) \left(\begin{array}{c} s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \epsilon \end{array} \right) \Rightarrow \\
& \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{cc} s_1 & \hat{o}_4 s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \text{vuole vedere la} & \epsilon \end{array} \right) \Rightarrow \\
& \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{cc} s_1 s_1^r \hat{o}_4^r \pi_3^r s_1 & \hat{o}_4 \\ \text{vuole vedere la} & \end{array} \right) \Rightarrow \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{cc} \hat{o}_4^r \pi_3^r s_1 & \hat{o}_4 \\ \text{vuole vedere la} & \end{array} \right) \Rightarrow \\
& \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{c} \hat{o}_4 \hat{o}_4^r \pi_3^r s_1 \\ \text{la vuole vedere} \end{array} \right) \Rightarrow \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{c} \pi_3^r s_1 \\ \text{la vuole vedere} \end{array} \right) \Rightarrow \\
& \left(\begin{array}{c} N_m \\ \text{Mario la vuole vedere} \end{array} \right) \Rightarrow \textcolor{yellow}{N_m \leq \pi_3} \\
& \left(\begin{array}{cc} \pi_3 & \pi_3^r s_1 \\ \text{Mario la vuole vedere} & \end{array} \right) \Rightarrow \left(\begin{array}{c} \pi_3 \pi_3^r s_1 \\ \text{Mario la vuole vedere} \end{array} \right) \Rightarrow \\
& \left(\begin{array}{c} s_1 \\ \text{Mario la vuole vedere} \end{array} \right)
\end{aligned}$$

We can also express the derivation in a graphical form as given below, with the two coordinates of one tuple depicted in red:



or by means of non-crossing underlinks:



Apart from being a very useful tool for the syntactic analysis and computation of natural language, tupled pregroup grammars can also handle semantic analysis. We illustrate the semantical interpretation for our two examples. Basic types are treated as functional or relational symbols and the iterated types determine the argument places. Single basic types - not introducing argument places - are translated by individual constants. It is possible to add a semantic translation for each lexical item. Then the translation of a sentence is computed by substitution according to the links (contractions).

In Examples 2 and 3 we now add translation to semantic types in the following way:

<i>Mario</i>	N_m	mario
<i>Dario</i>	N_m	dario
<i>vede</i>	$\pi_3^r s_1 o_4^l$	$\text{vedere}(x_1, x_2)$
<i>vuole</i>	$s_1 \bar{i}^l$	$\text{volere}(y)$
<i>la</i>	$\hat{o}_4$	la (within a tuple)
<i>vedere</i>	i	vedere
<i>vedere</i>	i	$\text{vedere}(\epsilon_-(\text{la}))$ (within a tuple)
ϵ	$s_1^r \hat{o}_4^r \pi_3^r s_1$	$\text{id}(z_1(z_3, z_2))$

Contractions according to underlinks define the substitutions. Thus in our examples we get:

$x_1 \rightarrow \text{mario}$
 $x_2 \rightarrow \text{dario}$
 $y \rightarrow \text{vedere}(\epsilon_-(\text{la}))$
 $z_1 \rightarrow \text{volere}(y)$
 $z_2 \rightarrow \text{la}$
 $z_3 \rightarrow \text{mario}$

Now we are able to compute the semantics of the sentence **Mario vede Dario** by substitution in the following way:

$\text{vedere}(x_1, x_2) = \text{vedere}(\text{mario}, \text{dario})$

Before we compute the semantics of the sentence **Mario la vuole vedere** we would like to emphasise the meaning of the tuple $\begin{pmatrix} i & \hat{o}_4 \\ \text{vedere} & \text{la} \end{pmatrix}$ (below in blue).

The type of the infinitive of the verb *vedere* [to see] should take as a complement the direct object in accusative case and the syntactic type of it should be io_4^l - but in the tuple the complement is already given (*la*). This fact is visible in the semantic types of both *la* and *vedere*, the first taking *la* as a direct object, the second treated as a trace of it.

mario	la	volere	y	vedere($\epsilon_-(\text{la})$)	z_1	z_2	z_3	id
Mario	la	vuole		vedere				ϵ
N_m	$\hat{o}_4$	$s_1 \bar{i}^l$		i	s_1^r	$\hat{o}_4^r$	π_3^r	$s_1 \Rightarrow s_1$

Now the meaning of the sentence is computed as follows:

$\text{id}(z_1(z_3, z_2)) = \text{id}(\text{volere}(y)(\text{mario}, \text{la})) = \text{id}(\text{volere}(\text{vedere}(\epsilon_-(\text{la})))(\text{mario}, \text{la})) =$
 $(\text{volere}(\text{vedere}(\epsilon_-(\text{la})))(\text{mario}, \text{la}))$

Here $\text{id}(x)=x$ is the identity function. The meaning of the sentence is read as follows: the function *volere(vedere)* [to want to see] takes two arguments - one for the subject and the other one for an object. In this case the direct object is a clitic which has to be placed before the verb phrase in the sentence and the tuple enables putting it in the right place with respect to the syntax. Normally the direct object is governed by the verb *vedere* [to see] (this is depicted by

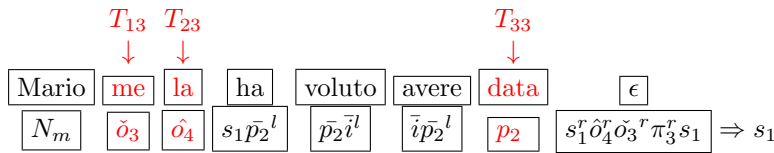
(vedere(ϵ _(la) which may be treated as a *trace*), however, in the present example the clitic pronoun in accusative case must take preverbal position and the modal verb *volere* [to want] is also involved.

Example 3

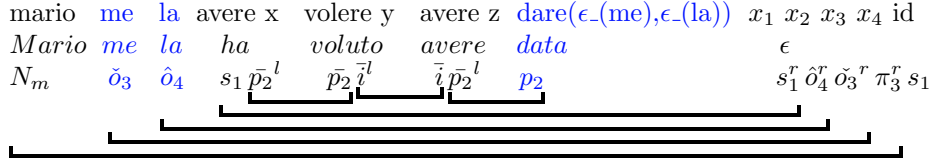
We now show how to justify for the correctness of the following sentence: **Mario me la ha voluto avere data.** ([Mario] [to me] [it] [has] [wished] [(to) have] [given] - Mario wished to have given it to me) The below given tuples are appropriate for the derivation. The attention should be given to a tuple with three coordinates, as the lexical entries are indispensable and should be taken together

$$\begin{aligned}
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} s_1 \bar{p}_2^l \\ ha \end{matrix} \right) \left(\begin{matrix} \bar{p}_2 \bar{i}^l \\ voluto \end{matrix} \right) \left(\begin{matrix} \bar{i} \bar{p}_2^l \\ avere \end{matrix} \right) \left(\begin{matrix} p_2 & \check{o}_3 & \hat{o}_4 \\ data & me & la \end{matrix} \right) \left(\begin{matrix} s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 \\ \epsilon \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} s_1 \bar{p}_2^l \\ ha \end{matrix} \right) \left(\begin{matrix} \bar{p}_2 \bar{i}^l \\ voluto \end{matrix} \right) \left(\begin{matrix} \bar{i} \bar{p}_2^l & p_2 & \check{o}_3 & \hat{o}_4 \\ avere & data & me & la \end{matrix} \right) \\
 & \left(\begin{matrix} s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 \\ \epsilon \end{matrix} \right) \Rightarrow \text{p}_2 \leq \bar{p}_2 \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} s_1 \bar{p}_2^l \\ ha \end{matrix} \right) \left(\begin{matrix} \bar{p}_2 \bar{i}^l \\ voluto \end{matrix} \right) \left(\begin{matrix} \bar{i} & \check{o}_3 & \hat{o}_4 \\ avere & data & me & la \end{matrix} \right) \left(\begin{matrix} s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 \\ \epsilon \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} s_1 \bar{p}_2^l \\ ha \end{matrix} \right) \left(\begin{matrix} \bar{p}_2 & \check{o}_3 & \hat{o}_4 \\ voluto & avere & data & me & la \end{matrix} \right) \left(\begin{matrix} s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 \\ \epsilon \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} s_1 & \check{o}_3 & \hat{o}_4 \\ ha & voluto & avere & data & me & la \end{matrix} \right) \left(\begin{matrix} s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 \\ \epsilon \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} s_1 s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 & \check{o}_3 & \hat{o}_4 \\ ha & voluto & avere & data & me & la \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 & \check{o}_3 & \hat{o}_4 \\ ha & voluto & avere & data & me & la \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} \hat{o}_4 \hat{o}_4^r \check{o}_3^r \pi_3^r s_1 & \check{o}_3 \\ la & ha & voluto & avere & data & me \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} \check{o}_3^r \pi_3^r s_1 & \check{o}_3 \\ la & ha & voluto & avere & data & me \end{matrix} \right) \Rightarrow \\
 & \left(\begin{matrix} N_m \\ Mario \end{matrix} \right) \left(\begin{matrix} \check{o}_3 \check{o}_3^r \pi_3^r s_1 \\ me & la & ha & voluto & avere & data \end{matrix} \right) \Rightarrow \dots \\
 & \left(\begin{matrix} s_1 \\ Mario & me & la & ha & voluto & avere & data \end{matrix} \right)
 \end{aligned}$$

Again, another way of presenting the sentence with three-coordinate tuple indispensable in our derivation:



and the same sentence presented with the non-crossing links with semantical types added:



<i>Mario</i>	N_n	mario	
<i>me</i>	$\check{o}_3$	me	(within a tuple)
<i>la</i>	$\hat{o}_4$	la	(within a tuple)
<i>ha</i>	$s_1 \bar{p}_2^l$	avere(x)	
<i>voluto</i>	$\bar{p}_2^l \bar{i}^l$	volere(y)	
<i>avere</i>	$\bar{i}^l p_2^l$	avere(z)	
<i>data</i>	p_2	dare($\epsilon_-(me), \epsilon_-(la)$) (within a tuple))	
ϵ	$s_1^r \hat{o}_4^r \check{o}_3^r \pi_3^r s_1$	id($x_1(x_4, x_3, x_2)$)	

The substitutions are as follows:

$x_1 \rightarrow \text{avere}(x)$
 $x_2 \rightarrow \text{la}$
 $x_3 \rightarrow \text{me}$
 $x_4 \rightarrow \text{mario}$
 $x \rightarrow \text{volere}(y)$
 $y \rightarrow \text{avere}(z)$
 $z \rightarrow \text{dare}(\epsilon_-(me), \epsilon_-(la))$

Now we are able to compute the semantics of the sentence **Mario me la ha voluto avere data** by substitution in the following way:

$\text{id}(x_1(x_4, x_3, x_2)) = \text{id}(\text{avere}(\text{volere}(\text{avere}(\text{dare}(\epsilon_-(me), \epsilon_-(la))))))(\text{mario}, \text{me}, \text{la}) =$
 $(\text{avere}(\text{volere}(\text{avere}(\text{dare}(\epsilon_-(me), \epsilon_-(la)))))(\text{mario}, \text{me}, \text{la})$

The interpretation of the meaning of the sentence is similar to the one given in Example 2, with *dare* [to give] being a ditransitive verb.

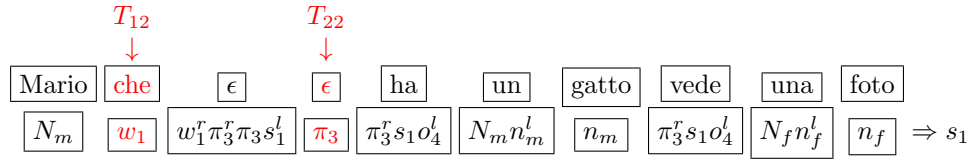
Example 4

We can justify for the correctness of the sentence with a relative pronoun **Mario che ha un gatto vede una foto** ([*Mario*] [*who*] [*has*] [*a*] [*cat*] [*looks*] [*at its*] [*picture*]) and proceed with the following tuples:

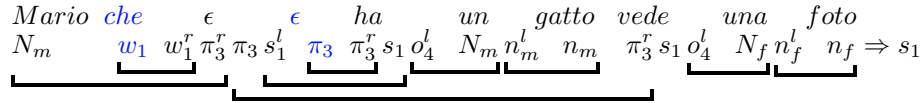
$$\begin{pmatrix} N_m \\ \text{Mario} \end{pmatrix} \begin{pmatrix} w_1 & n_m \\ \text{che} & \epsilon \end{pmatrix} \begin{pmatrix} w_1^r \pi_3^r \pi_3 s_1^l \\ \epsilon \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l \\ \text{ha} \end{pmatrix} \begin{pmatrix} N_m n_m^l \\ \text{un} \end{pmatrix} \begin{pmatrix} n_m \\ \text{gatto} \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 o_4^l \\ \text{vede} \end{pmatrix} \\
 \begin{pmatrix} N_f n_f^l \\ \text{una} \end{pmatrix} \begin{pmatrix} n_f \\ \text{foto} \end{pmatrix} \Rightarrow \dots \Rightarrow \begin{pmatrix} s_1 \\ \text{Mario che ha un gatto vede una foto} \end{pmatrix}$$

Another way of presenting the sentence with two-coordinate tuple indispensable

in our derivation below. Note that we make use of assumptions $N_m \leq o_4$ and $N_f \leq o_4$:



and the same sentence presented with the non-crossing links:



The meaning of the sentence, obtained in a similar way as in the former examples, gives us the following semantics:

$\text{vedere}((\text{mario}(\text{avere}(\text{X}, \text{un}(\text{gatto}))), \text{che}(\text{X}))), \text{una}(\text{foto}))$

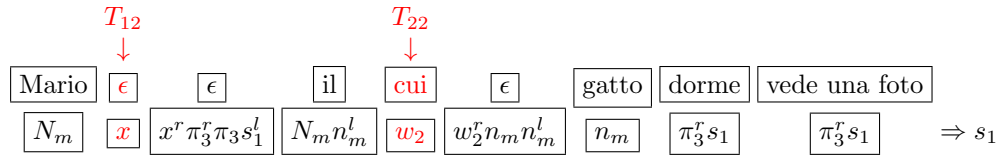
It can be seen that the function $\text{vedere}(\text{x}, \text{y})$ for *vede* [looks] takes two arguments (the subject and an object), and the subject (Mario) is modified by an appropriate relative clause.

Example 5

By means of tupled pregroup grammars we are also able to account for a sentencehood of the following string of words in Italian **Mario il cui gatto dorme vede una foto** ([Mario] [-] [whose] [cat] [is sleeping] [looks] [at its picture]). For that reason we need the following tuples from our lexicon:

$$\begin{aligned}
 & \left(\begin{array}{c} N_m \\ \text{Mario} \end{array} \right) \left(\begin{array}{c} N_m n_m^l \\ \text{il} \end{array} \right) \left(\begin{array}{c} w_2 \ x \\ \text{cui} \ \epsilon \end{array} \right) \left(\begin{array}{c} x^r \pi_3^r \pi_3 s_1^l \\ \epsilon \end{array} \right) \left(\begin{array}{c} w_2^r n_m n_m^l \\ \epsilon \end{array} \right) \left(\begin{array}{c} n_m \\ \text{gatto} \end{array} \right) \left(\begin{array}{c} \pi_3^r s_1 \\ \text{dorme} \end{array} \right) \\
 & \left(\begin{array}{c} \pi_3^r s_1 o_4^l \\ \text{vede} \end{array} \right) \left(\begin{array}{c} N_f n_f^l \\ \text{una} \end{array} \right) \left(\begin{array}{c} n_f \\ \text{foto} \end{array} \right) \Rightarrow \dots \\
 & \Rightarrow \left(\begin{array}{c} s_1 \\ \text{Mario il cui gatto dorme vede una foto} \end{array} \right)
 \end{aligned}$$

Below the same sentence presented with boxes:



and the same sentence presented with the non-crossing links:

[illegible]

Example 6

By means of tupled pregroup grammars we are also able to account for a sentencehood of the following strings of words in Italian (combining relative pronouns with clitics) **Mario che ha un gatto lo vuole guardare** (*[Mario] [who] [has] [a] [cat] [it] [wants] [to observe] Mario who has a cat wants to observe it*) or **Mario il cui gatto dorme lo vuole guardare** (*[Mario] [-] [whose] [cat] [is sleeping] [it] [wants] [to observe] - Mario whose cat is sleeping wants to observe it*). Let us conclude with the last sentence with the following lexicon elements:

$$\begin{aligned} & \begin{pmatrix} \pi_3 \\ \text{Mario} \end{pmatrix} \begin{pmatrix} N_m n_m^l \\ i_l \end{pmatrix} \begin{pmatrix} w_2 \ x \\ \text{cui} \ \epsilon \end{pmatrix} \begin{pmatrix} x^r \pi_3^r \pi_3 s_1^l \\ \epsilon \end{pmatrix} \begin{pmatrix} w_2^r n_m n_m^l \\ \epsilon \end{pmatrix} \begin{pmatrix} n_m \\ \text{gatto} \end{pmatrix} \begin{pmatrix} \pi_3^r s_1 \\ \text{dorme} \end{pmatrix} \\ & \begin{pmatrix} s_1 \bar{l}^l \\ \text{vuole} \end{pmatrix} \begin{pmatrix} i & \hat{o}_4 \\ \text{guardare} & \text{lo} \end{pmatrix} \begin{pmatrix} s_1^r \hat{o}_4^r \pi_3^r s_1 \\ \epsilon \end{pmatrix} \Rightarrow \dots \\ & \Rightarrow \begin{pmatrix} & s_1 \\ \text{Mario} & \text{il cui gatto dorme lo vuole guardare} \end{pmatrix} \end{aligned}$$

and, presented with boxes, the sentence looks as follows:

The diagram illustrates the structure of the input and output sequences for the model. The input sequence (top) consists of tokens: Mario, €, €, il, cui, €, gatto, dorme, ... The output sequence (bottom) consists of tokens: lo, vuole, guardare, €. The input sequence is divided into two parts by a red arrow labeled T_{12} . The output sequence is divided into two parts by a green arrow labeled T'_{12} . The input sequence is also divided into two parts by a red arrow labeled T_{22} . The output sequence is also divided into two parts by a green arrow labeled T'_{22} . The input sequence is also divided into two parts by a green arrow labeled T'_{22} . The output sequence is also divided into two parts by a red arrow labeled T_{22} .

4 Conclusions

In this paper examples involving both clitic pronouns and relative clauses have been considered, showing how the tuples introduced by this model of pregroup grammar can effectively constitute the *building blocks* of the grammatical structure of a natural language. The boxes presentations aim in fact at suggesting this idea of step-by-step syntactical analysis, while the underlink representations allow one to sketch the lines for a semantic analysis based on appropriate substitutions. Although the examples presented here are in a relatively small number, they are representative of two characteristic aspects of Italian grammar: preverbal clitic attachment and clitic climbing with modal auxiliaries, and cooccurrence of clitic pronouns with relative clauses.

References

1. Abrusci, M.: Classical Conservative Extensions of Lambek Calculus. *Studia Logica* 71, 277–314 (2002)
2. Bargelli, D., Lambek J.: An Algebraic Approach to French Sentence Structure, In De Groote, P., Morrill, G., Retoré, C. (eds.) *Logical Aspects of Computational Linguistics*. LACL 2009, Pages, 62–78. Springer, Berlin (2001)
3. Bargelli, D., Lambek, J.: An Algebraic Approach to Arabic Sentence Structure, *Linguistic Analysis*, Volume 31, Pages 301–315, (2001)
4. Béchet, D.: NP-completeness of grammars based upon products of free pregroups. In C. Casadio, B. Coecke, M. Moortgat, and P. Scott, editors, *Categories and Types in Logic, Language and Physics*, Festschrift on the occasion of Jim Lambek’s 90th birthday, LNCS 8222 51-62, Springer-Verlag (2014)
5. Béchet, D., Foret, A.: A pregroup toolbox for parsing and building grammars of natural languages, *Linguistic Analysis*, 36 (1-4), 473-482 (2010)
6. Buszkowski, W.: Lambek Grammars Based on Pregroups, in *Lecture Notes in Computer Science*, LACL 2009, Springer, 95-109 (2001)
7. Buszkowski, W.: Pregroups: Models and Grammars. In: *Relational Methods in Computer Science* LNCS 2561, 35-49 (2002)
8. Buszkowski, W., Lin, Z.: ‘Pregroup Grammars with Letter Promotions’, in *Lecture Notes in Computer Science*, LATA 6031, Springer, 130-141 (2010)
9. Casadio, C., Lambek, J.: An Algebraic Analysis of Clitic Pronouns in Italian, in A. Lecomte, F. Lamarche and G. Perrier (eds.), *Logical Aspects of Computational Linguistics*, LNAI 2099, Springer, 110-124 (2001)
10. Casadio, C., Lambek, J.: A Tale of Four Grammars. *Studia Logica* 71(2), 315–329 (2002)
11. Casadio, C.: Agreement and Cliticization in Italian: A Pregroup Analysis, *Lecture Notes in Computer Science*, LNCS 6031, Springer, 166–177 (2010)
12. Casadio, C., Kiślak-Malinowska, A.: Tupled Pregroups. A Study of Italian Clitic Patterns, *Proceedings LENLS 9*, ISBN 978-4-915905-51-3 (JSAI), 1–12 (2012)
13. Casadio, C., Kiślak-Malinowska, A.: Italian Clitic Patterns in Pregroup Grammar: State of the Art. *Categories and Types in Logic, Language, and Physics*, Festschrift on the occasion of Jim Lambek’s 90th birthday, LNCS 8222, Springer, 156-171 (2014)
14. Foret, A.: A modular and parameterized presentation of pregroup calculus. *Journal of Information and Computation* 208(5), 510–520 (2010).
15. Francez, N., Kaminski, M.: Commutation-Augmented Pregroup Grammars and Mildly Context-Sensitive Languages. *Studia Logica* 87(2/3), 295-321 (2007)
16. Kiślak-Malinowska, A.: Polish Language in Terms of Pregroups, in C. Casadio and J. Lambek (eds.), *Computational Algebraic Approaches to Natural Language*, Polimetrica, 145–172 (2008)
17. Kiślak-Malinowska, A.: Some Aspects of Polish Grammar in Terms of Tupled Pregroups, *Linguistic Analysis*, 93–119 (2010)
18. Kiślak-Malinowska, A.: Extended Pregroup Grammars Applied to Natural Languages, *Logic and Logical Philosophy*, Vol. 21, 229–252 (2012)
19. Lambek, J.: The Mathematics of Sentence Structure. *American Mathematics Monthly* 65, 154–169 (1958)
20. Lambek, J.: Type Grammars Revisited, in A. Lecomte, F. Lamarche and G. Perrier (eds.), *Logical Aspects of Computational Linguistics*, LNAI 1582, Springer, Berlin, 1-27, (1999)
21. Lambek, J.: *From Word to Sentence*, Polimetrica, (2008)

22. Lambek, J.: Compact Monoidal Categories from Linguistics to Physics, In: New Structures for Physics, Coecke, B. (ed.): *Lecture Notes in Physics*, Springer-Verlag, Vol 813, 467–487, (2011)
23. Monachesi, P.: A lexical Approach to Italian Cliticization, *Lecture Notes 84*, CSLI, Stanford (1999).
24. Moortgat, M.: Categorical Type Logics. In: van Benthem, J., ter Meulen, A. (eds.) *Handbook of Logic and Language*, Elsevier, Amsterdam, 93–177 (1997)
25. Moortgat, M.: Symmetric Categorical Grammar. *Journal of Philosophical Logic*, 38, 681–710 (2009)
26. Moot, R., Retoré, C.: Les indices pronominaux du français dans les grammaires catégorielles, *Linguisticae Investigationes*, Vol 29, 1, 137–146 (2006)
27. Preller, A., Category Theoretical Semantics for Pregroup Grammars, *Logical Aspects of Computational Linguistics*, LNCS 3492, 238–254 (2005)
28. Preller, A., Lambek, J.: Free Compact 2-Categories, *Mathematical Structures in Computer Science*, Vol 17, 309–40 (2007).
29. Preller, A., Sadrzadeh, M.: Semantic Vector Models and Functional Models for Pregroup Grammars, *Journal of Logic, Language and Information* 20 (4), 419–443 (2011)
30. Preller, A., Degeilh, S.: Efficiency of Pregroups and the French Noun Phrase, *Journal of Logic, Language and Information*, 14, 423–444 (2005)
31. Rizzi, L.: Issues in Italian Syntax. *Foris Publications*, Dordrecht, 1982; second printing: Mouton de Gruyter, The Hague (1993)
32. Sadrzadeh, M.: Pregroup Analysis of Persian Sentences, in Casadio, C., Lambek, J. (eds.): *Recent Computational Algebraic Approaches to Morphology and Syntax*, Polimetria, Milan (2008)
33. Stabler, E.: Tupled pregroup grammars, in C. Casadio and J. Lambek (eds.), *Computational Algebraic Approaches to Natural Language*, Polimetria, 23–52 (2008)