Pregroup grammars with linear parsing of the French verb phrase

Anne Preller, Violaine Prince

▶ **To cite this version:**

Anne Preller, Violaine Prince. Pregroup grammars with linear parsing of the French verb phrase. Claudia Casadio, Joachim Lambek. Computational Algebraic Approaches to Natural Language, Polimetrica, Milano-Italy, pp.53-84, 2008, 978-88-7699-125-7, 978-88-7699-126-4. lirmm-00306504

HAL Id: lirmm-00306504

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00306504

Submitted on 27 Jul 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

C. Casadio and J. Lambek (eds), *Computational algebraic approaches to natural language*, 53-84 © 2008 Polimetrica International Scientific Publisher Monza/Italy

Pregroup grammars with linear parsing of the French verb phrase

Anne Preller and Violaine Prince

LIRMM
Montpellier - France

Abstract. The parsing algorithm for pregroup grammars presented here exploits regularities of types of a pregroup grammar. Sufficient conditions are given for the algorithm to be linear and complete. Its working is illustrated by a grammar which handles unbounded dependencies and agreement with clitics in French. The grammar includes control of the complement of the infinitive by modal and causal verbs.
Keywords: categorial grammars, pregroup grammars, linear parsing algorithm, distant dependencies, complement control, agreement of reflexive pronouns and clitics in French.

1. Introduction

Pregroup grammars belong to the family of categorial grammars and were introduced by [6]. A pregroup grammar consists of a dictionary and only one rule: generalized contraction. A dictionary is a list of ordered pairs $v : X$, called *lexical entries*, where v is a word of the language and X an element of the pregroup, called *type*. The same word may be listed several times, but with different types. To analyze a string of words $v_1 \dots v_n$ one chooses types X_i such that $v_i : X_i$ belongs to the dictionary for $1 \leq i \leq n$ and checks if successive applications of the generalized contraction rule reduce the concatenated type $X_1 \dots X_n$ to the *sentence type* $\boldsymbol{s}$. Each such choice of contractions is a *reduction from*

$X_1 \dots X_n$ *to* $\boldsymbol{s}$. A string of words $v_1 \dots v_n$ is recognized as a sentence, if the type $X_1 \dots X_n$ reduces to $\boldsymbol{s}$ for some type assignment for $v_1 \dots v_n$. Each such reduction constitutes a parsing of $v_1 \dots v_n$.

Pregroup grammars have parsing algorithms which run in cubic time, i.e. in time proportional to n^3, where n is the length of the string of words. Hence any endeavor to improve complexity will aim at linear run-time algorithms.

Ambiguity enters pregroup grammars in two ways: I) A string of words may have multiple type assignments. II) Sentences may have up to 2^n parsings for one and the same type assignment. As an example, consider the rigid dictionary listing three words, each with a single type, namely $u: \ \boldsymbol{s}a^\ell, v: \ aa^\ell$ and $w: \ aa^r a$ respectively. Then sentences of he form $uv^n w$ have at least 2^n different reductions to the sentence type. Hence an efficient parsing algorithm cannot construct all parsings of a sentence during its run-time. What we require of such an algorithm is to decide whether or not a string of words is a sentence and, if it is one, to produce one of the possible parsings.

Ambiguity of the second kind typically arises with relative clauses or coordination. It alone can be responsible for the cubic time bound. However, ambiguity of the second kind need not be an obstacle to linear parsing. Indeed, these linguistic phenomena can be handled by a class of pregroup grammars with complete linear parsing algorithms for every given type assignment, see [12]. Here, we show that the same holds for ambiguity of the first kind. We define criteria that make it possible to recognize type assignments with no reduction to the sentence type at an early stage during processing. Hence, the algorithms of [*loc.cit.*] can be adapted to recognize and avoid the 'losing' type assignments, processing thus only a constant number of type assignments.

Here we address ambiguity of the first kind. The abundance of multiple type assignments in our grammar is partly due to clitics depending on features like person, gender and number without expressing them morphologically. Another reason are dependencies involving auxiliaries and the past participle as well as modal and causal verbs and the infinitive. The multiple entries in the dictionary are the price to pay to make the pregroup grammar handle such dependencies both syntactically and semantically without additional structure. The semantic interpretation in predicate logic of the grammar is out of the scope of this paper. Examples from the grammar presented here are used in [11] for a study of such an interpretation. In this paper, we limit ourselves to a less formal description of dependency.

The next section recalls some basic properties and results concerning pregroup grammars. The following Section 3 presents a minidictionary for the French verb phrase. Reflexive pronouns, combinations of the direct and indirect object clitics and their agreement with the past participle are included . Such a dictionary is necessarily incomplete. It would have to be completed in many aspects for an application, but we think that this can be done on the foundations given here. In the final Section (4), we study the algorithm and the criteria which make it recognize 'losers' and skip them early on. A 'loser' is a partial type assignment which cannot be extended to one with a reduction to the sentence type. Hence the algorithm parses only non-losing type assignments, and each in linear time. Finally, we mention sufficient conditions for the existence on a bound on the number of non-losing type assignments. In this case the algorithm produces a parsing for each of the type assignments with a reduction to the sentence type.

2. Fundamental properties

We recall the definition of pregroups, the construction of a freely generated pregroup by [6] and the geometrical characterization of derivations in pregroup grammars as graphs of underlinks.

A preordered monoid $< P, 1, \cdot, \rightarrow >$ is a set P with a distinguished element 1, 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 $b \rightarrow c$ implies $a \rightarrow c$
$a \rightarrow b$ implies $u \cdot a \cdot v \rightarrow u \cdot b \cdot v$.

The dot denoting multiplication is generally omitted. As usual, we say that two elements a and b are *non-comparable* if $a \not\rightarrow b$ and $b \not\rightarrow a$.

A pregroup is a 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. $a \to b$ if and only if $ab^r \to 1$ if and only if $b^\ell a \to 1$.

The *free* pregroup $P(B)$ generated by a partially ordered set of *basic types* B is characterized in [6] as the free monoid generated from the set of *simple types* Σ consisting of the *iterated adjoints* of basic types

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

We call a the *base* and z the *iterator* of the simple type $t = a^{(z)}$. The basic types $a \in B$ are identified with $a^{(0)} \in \Sigma$ and therefore included in the simple types. Elements of $P(B)$ are called *types*, they are 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, for a basic type a we have

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

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

(Induced step)	$Xa^{(z)}Y \to Xb^{(z)}Y$
(Generalized contraction)	$Xa^{(z)}b^{(z+1)}Y \to XY$
(Generalized expansion)	$Y \to Xa^{(z+1)}b^{(z)}Y$

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 $b \to a$.

In linguistic applications, the relevant inequalities have the form

$$t_1 \dots t_m \to \boldsymbol{s} ,$$

where the t_i's are simple and $\boldsymbol{s}$ is a basic type. A derivation of such an inequality can be obtained by generalized contractions and induced steps only, see Proposition 2 of [6]. For example, consider the dictionary

Marie	:	$\nu_{\mathtt{fs}}$
Jean	:	$\nu_{\mathtt{ms}}$
examine	:	$\pi^r_{3\mathtt{s}} \boldsymbol{s} o^\ell$

.

The basic type $\nu_{\mathtt{fs}}$ stands for 'proper name feminine singular' and similarly $\nu_{\mathtt{ms}}$ stands for 'proper name masculine singular'. We also have the basic type $\pi_{3\mathtt{s}}$ for the 'subject third person singular'. The feature *gender* has two values, namely `masculine`, abbreviated by `m`, and `feminine`, abbreviated by `f`. Similarly, the feature *number* may take the values `s` (= `singular`) and `p` (= `plural`). The basic types o and $\boldsymbol{s}$ stand for 'direct object' respectively for 'sentence in the present'. It is assumed that $\nu_{gn} \to \pi_{3n}$ and $\nu_{gn} \to o$ for $g = \mathtt{m}, \mathtt{f}$ and $n = \mathtt{s}, \mathtt{p}$.

To analyze a string of words, concatenate the types from the dictionary in the order of the words. The string of words is reputed a sentence if and only if the concatenated type has a derivation to the sentence type. For example,

$$\begin{array}{c} \textit{Marie} \quad \textit{examine} \quad \textit{Jean} \\ (\textsc{Mary examines John}) \\ \underbrace{(\nu_{\mathtt{fs}})}\ \ \underbrace{(\pi^r_{3\mathtt{s}}}\ \boldsymbol{s}\ \underbrace{o^\ell)}\ \ \underbrace{(\nu_{\mathtt{ms}})} \quad \to \boldsymbol{s} \end{array}$$

This derivation is justified by the generalized contractions $\nu_{\mathtt{fs}}\pi^r_{3\mathtt{s}} \to 1$ and $o^\ell \nu_{\mathtt{ms}} \to 1$. As customary, the types have been written under the words and the generalized contractions are indicated by underlinks.

The underlinks uniquely determine the derivation of a string of simple types $s_1 \dots s_n$ to the substring $s_{i_1} \dots s_{i_p}$ consisting of generalized contractions only. Indeed, such a derivation is determined by a graph R, called *reduction*, and a set of *algebraic conditions*.
The *reduction* R is a set of two-element subsets $\{i,k\} \subseteq \{1,\dots,n\}$, called *underlinks*, and satisfies

- if $i \neq i_l$ for $1 \leq l \leq p$, there is exactly one k such that $\{i,k\} \in R$,
- if $\{i,k\} \in R$ then $i \neq i_l$ for all $l \in \{1,\dots,p\}$,
- if $\{i,k\} \in R$ and $i < j < k$ then there is $i < m < k$ such that $\{j,m\} \in R$.

The algebraic conditions are the generalized contractions

- if $i < k$ and $\{i,k\} \in R$ then $s_i s_k \to 1$

We say that a reduction R *reduces* (equivalently, *is a transition from*) $s_1 \dots s_n$ *to* $s_{i_1} \dots s_{i_p}$, in symbols $R : s_1 \dots s_n \Rightarrow s_{i_1} \dots s_{i_p}$, if all four conditions above hold.

The term *transition* has been introduced in [9] for a geometric description of an arbitrary derivation. In linguistic applications, where only derivations to substrings are considered, it is more common to use the term *reduction*.

Representing an underlink $\{i,k\} \in R$ with $i < k$ as an edge between the horizontally aligned vertices

$$s_1 \dots \underbrace{s_i \dots s_k} \dots s_n \ ,$$

we can describe R as a planar graph. It is non-directed, has no loops, its vertices are linearly ordered and labeled by simple types. Its edges do not cross and have their endpoints in $\{1,\dots,n\} - \{i_1,\dots,i_p\}$. Moreover, a vertex different from $i_1,\dots,i_p$ is endpoint of exactly one edge.

If the substring $s_{i_1} \dots s_{i_p}$ cannot be contracted any further, it is called an *irreducible form* of $s_1 \dots s_n$. The empty string 1 and every simple type is irreducible. A string of simple types has at least one irreducible form, but their may be more than one, for example $a^\ell a a^r a^{rr} \to 1$ and $a^\ell a a^r a^{rr} \to a^\ell a^{rr}$. Even for a fixed irreducible form, there may be different reductions bringing the type to that form, e.g. both

$$\underbrace{a^\ell \underbrace{a \underbrace{a^\ell a}\ a^r}\ a} \quad \text{and} \quad \underbrace{a^\ell a}\ \underbrace{a^\ell \underbrace{a a^r}\ a}$$

are reductions to the empty string. The multiple reductions are due to the presence of *critical triples* like $a^\ell a a^r$ or more generally stu such that $st \to 1$ and $tu \to 1$. They cause ambiguities of kind II and correspond to different semantic interpretations. In general, they increase computational complexity.

Finally, the same reduction may reduce quite different types to a given string. For example, the reduction $\{\{1,2\},\{3,6\},\{4,5\}\}$ reduces the three different types below to the empty string

$$\underbrace{a^\ell a}\ \underbrace{a^\ell \underbrace{a\, a^r}\ a}, \quad \underbrace{a^\ell a}\ \underbrace{b^\ell \underbrace{c\, c^r}\ b} \quad \text{and} \quad \underbrace{b^{\ell\ell\ell} b^{\ell\ell}}\ \underbrace{d^r \underbrace{c\, c^r}\ d^{rr}} \ .$$

These geometrically identical reductions may have the same semantical interpretations. It suffices to translate corresponding types by the same semantic type, for example the type a in the first grammar and the type $b^{\ell\ell}$ of the third grammar *etc.* Differences caused just by names of basic types occur in the study of syntactic agreement. They can be used to speed up the parsing algorithm described in Section (4). Differences

between the third and the first two types show that the number of iterations of adjoints does not really matter.

The algorithm discussed in Section 4 searches for a reduction to the empty string. It follows immediately from the definitions that such an algorithm also solves the problem for a reduction to the sentence type. For later reference, we formulate this as the following fact.

Fact 1. *There is a one-one correspondence between reductions* $R : T \Rightarrow \boldsymbol{s}$ *and reductions* $R^* : T\boldsymbol{s}^r \Rightarrow 1$ *where the latter is obtained from the former by adding a link through the only position of* T *not linked by* R *and the last position in* $T\boldsymbol{s}^r$ *.*

The problem addressed in this paper is how to avoid processing type assignments which have no reduction to the sentence type. Two things influence the complexity of computation: the form of the types in the dictionary and the order on the set of the basic types.

The sample dictionary of Section (3) uses only basic types or right or left adjoints of basic types. Moreover, the connected components of the basic types are trees. these simplifications do not diminish expressivity. This follows from the next two facts.

Definition 1. *Two pregroup grammars (or their dictionaries) are strongly equivalent, if their dictionaries* D *and* D' *list the same words and if for every string of words* $v_1 \dots v_n$ *and every type assignment* $T_1, \dots, T_n$ *for* $v_1 \dots v_n$ *from one of the dictionaries, say* D*, and every reduction* R *which reduces* $T_1 \dots T_n$ *to the sentence type there is a type assignment* $T'_1, \dots, T'_n$ *for* $v_1 \dots v_n$ *from* D' *such that* R *also reduces* $T'_1 \dots T'_n$ *to the sentence type. In particular,* T_i *and* T'_i *have the same length for* $1 \leq i \leq n$ *.*

Obviously, strongly equivalent dictionaries have the same sentences, the same parsings and the same semantic interpretations.

Fact 2 ([10])**.** *Every pregroup grammar is strongly equivalent to one that uses basic types and left and right adjoint of basic types only. Moreover, the dictionary of the latter can be effectively computed.*

The order of the set of basic types can also be simplified without changing expressivity. This result has been presented by the first author at a workshop in Chieti in May 2005. A detailed proof has circulated since, but not been published. We reproduce it here, because we use it 'backward' when designing our dictionary in the next section.

Fact 3. *Let $B = (\mathtt{B}, \rightarrow)$ be a partially ordered set and $B' = (\mathtt{B}, =)$ the same set ordered by equality. Then for every dictionary D over B, there is a strongly equivalent dictionary D' over B' satisfying for every word v*

$$D(v) \subset D'(v)\,.$$

Moreover, for every $X' = s'_1 \dots s'_n \in D'(v)$ there is a type of the same length $X = s_1 \dots s_n$ in $D(v)$ for which s'_i and s_i have the same iterator, for $1 \leq i \leq n$.

Proof: The idea is to replace $a^{(2z-1)}$ by $b^{(2z-1)}$ for $b \rightarrow a$. Let $X = s_1 \dots s_m \in D(v)$ be given. We form the set of types

$$D_X = \left\{ s'_1 \dots s'_m : \forall i (s'_i = s_i \vee \exists a \exists b (a \rightarrow b \wedge s'_i = b^{(2z-1)} \wedge s_i = a^{(2z-1)})) \right\}$$

Note that all $X' \in D_X$ have the same length and the same sequence of iterators as X and that $s_i \rightarrow s'_i$ for $1 \leq i \leq m$. Moreover, $X \in D_X$. The set of types $D'(v)$ assigned to v by the new dictionary is

$D'(v) = \bigcup_{X \in D(v)} D_X$

Suppose that $v_1 \dots v_n$ is a s-sentence of D. Consider a reduction R such that $R : X_1 \dots X_n s^r \Rightarrow 1$ in $P(B)$. We must show that there is a type assignment $X'_1 \dots X'_n$ for $v_1 \dots v_n$ from D' such that the reduction R satisfies $R : X'_1 \dots X'_n s^r \Rightarrow 1$ in $P(B')$. Express $X_1 \dots X_n s^r$ as a string of simple types $s_1 \dots s_m$. Consider any underlink

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

Recall that $s_i s_k \rightarrow 1$ if and only if $s_k \rightarrow s_i^r$ if and only if $s_i \rightarrow s_k^\ell$ in $P(B)$.

Case 1: the iterator of s_i is odd.

Then there are $b \rightarrow a$ in B such that $s_i = a^{(2z-1)}$ and $s_k = b^{(2z)}$. Let $s'_i = b^{(2z-1)}$ and $s'_k = s_k$.

Case 2: The iterator of s_i is even. Then the iterator of s_k is odd and there are $b \rightarrow a$ in $P(B)$ such that $s_k = a^{(2z-1)}$ and $s_i = b^{(2z-2)}$. Let $s'_i = s_i$ and $s'_k = b^{(2z-1)}$.

In both cases $s'_i s'_k \rightarrow 1$ in $P(B')$. We obtain thus a new type assignment $X'_1 \dots X'_n = s'_1 \dots s'_m$ with $X'_j \in D'(v_j)$, for $1 \leq j \leq n$. Clearly, $X'_1 \dots X'_n$ is a type assignment for $v_1 \dots v_n$ from D' for which the reduction R satisfies $R : X'_1 \dots X'_n s^r \Rightarrow 1$ in $P(B')$.

Conversely, let $v_1 \dots v_n$ be a s-sentence of D' and $R' : X'_1 \dots X'_n s^r \Rightarrow 1$ in $P(B')$ with $X'_j \in D'(v_j)$, for $1 \leq j \leq n$. Every $X'_j = s'_{j1} \dots s'_{jm_j} \in D'(v_j)$ is obtained from some type $X_j = s_{j1} \dots s_{jm_j} \in D(v_j)$ such that

$s_{jp} \to s'_{jp}$ for $1 \leq p \leq m_j$. Hence, from $s'_i s'_k \to 1$ follows $s_i s_k \to 1$ and therefore R' satisfies $R' : X_1 \dots X_n s^r \Rightarrow 1$ in $P(B)$.

If the basic types are discretely ordered a type checking algorithm will execute the test $a^{(z)} b^{w)} \to 1$ faster, because the algorithm just checks if $a = b$ and $w = z + 1$ without consulting the table listing the ordered pairs of B. The main advantage of a proper partial order is to diminish the size of the dictionary. The dictionary of the next section uses the proof above 'backward'. Discrete subsets of basic type are collected into supertypes that occur only as left or right adjoints in lexical entries. There are more type assignments in the equivalent discrete dictionary, but the number of those with a reduction to the sentence type is the same. We show in Section 4 on parsing that run-time does not increase for a rather expressive class of dictionaries.

3. The dictionary

The grammar for clitics in French given here differs from that in [1] in several aspects. One is that we included agreement of gender and number of the direct object clitics and also agreement of person, if the pronoun is reflexive. Another one is that we avoid the meta-rule of [*loc.cit.*] and stay within an ordinary pregroup grammar. Moreover, our grammar is conceived for computability by linear algorithms and for easy proofs of that fact. Finally, we wanted a grammar that can be interpreted in two-sorted predicate logic.

Recall the lexical entries given in the previous section.

$$\begin{array}{ll} \textit{Marie} & : \nu_{\mathtt{fs}} \\ \textit{Jean} & : \nu_{\mathtt{ms}} \\ \textit{examine} & : \pi^r_{3_{\mathtt{s}}} \boldsymbol{s} o^\ell \end{array} \tag{1.1}$$

As we want to include the personal pronouns *je*, *tu*, *il*, *elle, nous, vous, ils elles*, we add the basic types π_{pgn} for the subject of person p, gender g and number n. We postulate

$$\pi_{pgn} \to \pi_{pn}, \text{ for } p = 1, 2, 3\,,\, g = \mathtt{m}, \mathtt{f}\,,\, n = \mathtt{s}, \mathtt{p}\,.$$

The first and second person pronouns *je*, *te*, *nous* and *vous* can be both masculine and feminine. Moreover, the plurals *nous* and *vous* can occur as subject, direct or indirect object, be reflexive and non-reflexive. We

introduce specific types for these pronouns so as to diminish the number of type assignments.

$$\begin{array}{l} \textit{nous} : \nu_{1g\mathtt{p}} \text{ for } g = \mathtt{m},\ \mathtt{f} \\ \textit{vous} : \nu_{2g\mathtt{p}} \text{ for } g = \mathtt{m},\ \mathtt{f}\,. \end{array} \tag{1.2}$$

We express that they can be subject by postulating

$$\nu_{pgn} \to \pi_{pgn} \text{ for } p = 1, 2\,,\ g = \mathtt{m}, \mathtt{f} \text{ and } \ n = \mathtt{s},\ \mathtt{p}\,.$$

Other entries in the dictionary are

$$\begin{array}{ll} \textit{je} & : \pi_{1g\mathtt{s}} \text{ for } g = \mathtt{m}, \mathtt{f} \\ \textit{tu} & : \pi_{2g\mathtt{s}} \text{ for } g = \mathtt{m}, \mathtt{f} \\ \textit{il} & : \pi_{3\mathtt{ms}} \\ \textit{elle} & : \pi_{3\mathtt{fs}} \\ \textit{ils} & : \pi_{3\mathtt{mp}} \\ \textit{elles} & : \pi_{3\mathtt{fp}} \end{array} \tag{1.3}$$

Then we find the reduction

$$\begin{array}{cccc} \textit{Nous} & \textit{examinons} & \textit{Jean} & \\ \text{WE} & \text{EXAMINE} & \text{JEAN} & \\ \underbrace{(\nu_{1g\mathtt{p}})} & \underbrace{(\pi^r_{1\mathtt{p}}\ \boldsymbol{s}\ o^\ell)} & \underbrace{(\nu_{\mathtt{ms}})} & , \end{array}$$

which corresponds to two distinct type assignments, namely $g = \mathtt{m}, \mathtt{f}$.

3.1. Clitics and the simple tenses

We consider preverbal direct and indirect object clitics in French. In simple tenses there is no agreement with the object clitic except for the reflexive pronouns, which agree with the subject. We discuss shortly the distant dependency and anaphoric content of the reflexive pronoun including the case when it is separated from the subject by modal verbs.

3.1.1. Non-reflexive clitics

We add the basic types

$$o_{pgn} \text{ for } p = 1, 2, 3\,, g = \mathtt{m}, \mathtt{f} \text{ and } n = \mathtt{s}, \mathtt{p}\,,$$

for direct object clitics. We invent a 'helper' type $\bar{o}$ which does not depend on person, gender and number and postulate

$$o_{pgn} \to \bar{o} \text{ for } p \in \{1,2,3\}\ ,\ g \in \{\mathtt{m},\mathtt{f}\}\ ,\ n \in \{\mathtt{s},\mathtt{p}\}\ .$$

We express that *nous* and *vous* can be a direct object by postulating

$$\nu_{1g\mathtt{p}} \to o_{1g\mathtt{p}} \text{ and } \nu_{2g\mathtt{p}} \to o_{2g\mathtt{p}} \text{ for } g = \mathtt{m},\mathtt{f}\ .$$

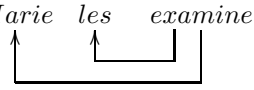
The lexical entries for preverbal direct object clitics and transitive verbs are[1]

$$\begin{array}{lll} \textit{examine} & : \bar{o}^r \pi^r_{3\mathtt{s}} \boldsymbol{s} & \\ m' & : o_{1g\mathtt{s}} & \text{for } g = \mathtt{m},\mathtt{f} \\ t' & : o_{2g\mathtt{s}} & \text{for } g = \mathtt{m},\mathtt{f} \\ l' & : o_{3g\mathtt{s}} & \text{for } g = \mathtt{m},\mathtt{f} \\ les & : o_{3g\mathtt{p}} & \text{for } g = \mathtt{m},\mathtt{f}\quad . \end{array} \tag{1.4}$$

Then reductions to the sentence type below provide the expected analysis, namely

$$\begin{array}{ll} \textit{Marie}\ \ \textit{les}\ \ \textit{examine} & \textit{elle}\ \ \textit{nous}\ \ \textit{examine} \\ (\text{MARY THEM EXAMINES}) & (\text{SHE US EXAMINES}) \\ (\nu_{\mathtt{fs}})\ \underbrace{(o_{3g\mathtt{p}})\ \underbrace{(\bar{o}^r \pi^r_{3\mathtt{s}}}\ \boldsymbol{s})} & \underbrace{(\pi_{\mathtt{fs}})\ \underbrace{(\nu_{1g\mathtt{p}})\ \underbrace{(\bar{o}^r \pi^r_{3\mathtt{s}}}\ \boldsymbol{s})}}\ ,\ \text{for } g = \mathtt{m},\mathtt{f}\ . \end{array}$$

Note that the links of the reduction correctly capture the dependencies in predicate logic. Indeed, an underlink, when oriented from a left or right adjoint to a basic type, expresses dependency. In the left-hand graph, the binary predicate `examiner` corresponding to the transitive verb *examiner* depends on two distinct entities *Marie* and *les*.



The indirect object clitics do not depend on the gender and do not require agreement with the past participle. Hence we introduce

$$\omega_{pn} \text{ with } p = 1,2,3 \text{ and } n = \mathtt{s},\mathtt{p}\ .$$

[1] We consider only the short forms *m', t' l'* which are used if the following word starts with a vowel. The full forms *me*, *te*, *le* and *la* are required if the next word starts with a consonant. The grammar can distinguish between the two if the type of clitics depends on the feature 'vowel' with two values, `yes`, `no`, but we will ignore this here.

Again, we may lower the number of entries for the bi-transitive verbs by inventing 'helpers' $\bar{o}_p$, $\bar{\omega}_p$ and $\bar{\omega}$, for which we postulate

$$o_{pgn} \to \bar{o}_p\,,\ \omega_{pn} \to \bar{\omega}\,,\ \omega_{pn} \to \bar{\omega}_p\,,\quad \text{for } p = 1,2,3\,, g = \mathtt{m}, \mathtt{f} \text{ and } n = \mathtt{s}, \mathtt{p}\,.$$

The possibility for *nous* and *vous* to occur as indirect objects is guaranteed by

$$\nu_{1g\mathtt{p}} \to \omega_{1\mathtt{p}} \text{ and } \nu_{2g\mathtt{p}} \to \omega_{2\mathtt{p}} \text{ for } g = \mathtt{m}, \mathtt{f}\,.$$

For the other clitics and verbs with an indirect object, we add the entries

$$\begin{array}{lll} \textit{parles} & : \bar{\omega}^r \pi^r_{2\mathtt{s}} \boldsymbol{s} & \\ \textit{donne} & : \bar{\omega}^r_3 \bar{o}^r_3 \pi^r_{3\mathtt{s}} \boldsymbol{s} & \\ \textit{donne} & : \bar{o}^r_3 \bar{\omega}^r_p \pi^r_{3\mathtt{s}} \boldsymbol{s} & \text{for } p = 1,2 \\ \textit{me} & : \omega_{1\mathtt{s}} & \\ \textit{te} & : \omega_{2\mathtt{s}} & \\ \textit{lui} & : \omega_{3\mathtt{s}} & \\ \textit{leurs} & : \omega_{3\mathtt{p}} & . \end{array} \tag{1.5}$$

The helper types depend on the person, because in grammatical combinations of the direct and indirect object clitics one of them must be third person. The order also depends on the persons, for example *les leurs*, *me les* and *le lui*, but not **leurs les*, **me lui*, **lui la* etc. .

A few examples of reductions to the sentence type are

$$\underset{(\pi_{2g\mathtt{s}})}{\textit{tu}}\ \underset{(\nu_{1g'\mathtt{p}})}{\textit{nous}}\ \underset{(\bar{\omega}^r\ \pi^r_{2\mathtt{s}}\ \ \boldsymbol{s})}{\textit{parles}} \qquad \underset{(\nu_{\mathtt{ms}})}{\textit{Jean}}\ \underset{(o_{3g\mathtt{p}})}{\textit{les}}\ \underset{(\omega_{3\mathtt{p}})}{\textit{leurs}}\ \underset{(\bar{\omega}^r_3\ \bar{o}^r_3\ \pi^r_{3\mathtt{s}}\ \ \boldsymbol{s})}{\textit{donne}}\,,$$

$g = \mathtt{m}, \mathtt{f},\ g' = \mathtt{m}, \mathtt{f}$.

Note that on the whole there are six type assignments with a reduction to the sentence type above. As they only differ by the gender value for the personal pronouns, but not by the underlinks, we can compute them just once. The expedient is not to assign values to g and g', as they are irrelevant for the reduction itself.

3.1.2. Reflexive clitics

The reflexive pronouns agree in person, gender and number with the subject. They also have an anaphoric content by identifying the object with the subject. We therefore add the *dummy* types for direct and indirect objects

$$\hat{o}_{pgn} \text{ and } \hat{\omega}_p,\ p = 1,2,3\,, g = \mathtt{m}, \mathtt{f} \text{ and } n = \mathtt{s}, \mathtt{p}\,.$$

As for the non-reflexive clitics, we require 'helpers' $\hat{o}_{gn}$, $\hat{o}_p$ and $\hat{o}$ for which we postulate

$$\hat{o}_{pgn} \to \hat{o}_{gn}\,,\ \hat{o}_{pgn} \to \hat{o}_p\,,\ \hat{o}_{pgn} \to \hat{o}\,,\ \hat{\omega}_p \to \hat{\omega} \text{ for } p = 1,2,3,\ g = \mathtt{m}, \mathtt{f},\ n = \mathtt{s}, \mathtt{p}\,.$$

Besides the lexical entries for the reflexive pronouns, we also need new types for the verbs. In the bi-transitive case the reflexive clitic is always the indirect object and precedes the direct object clitic. For example

$$\begin{array}{lcll}
\textit{s'} & : & \pi^r_{3gn}\pi_{3gn}\hat{o}_{gn} & \text{for } g \in \{\mathtt{m},\mathtt{f}\}\,,\ n \in \{\mathtt{s},\mathtt{p}\} \\
\textit{s'} & : & \pi^r_{3gn}\pi_{3gn}\hat{\omega}_3 & \text{for } g \in \{\mathtt{m},\mathtt{f}\}\,,\ n \in \{\mathtt{s},\mathtt{p}\} \\
\textit{nous} & : & \pi^r_{1g\mathtt{p}}\pi_{1g\mathtt{p}}\hat{o}_{g\mathtt{p}} & \text{for } g \in \{\mathtt{m},\mathtt{f}\} \\
\textit{nous} & : & \pi^r_{1g\mathtt{p}}\pi_{1g\mathtt{p}}\hat{\omega}_1 & \text{for } g \in \{\mathtt{m},\mathtt{f}\} \\
\textit{examine} & : & \hat{o}^r\pi^r_{3\mathtt{s}}\boldsymbol{s} & \\
\textit{examinons} & : & \hat{o}^r\pi^r_{1\mathtt{p}}\boldsymbol{s} & \\
\textit{donnons} & : & \bar{o}^r_3\hat{\omega}^r_1\pi^r_{1\mathtt{p}}\boldsymbol{s} & \\
\textit{donne} & : & \bar{o}^r_3\hat{\omega}^r_3\pi^r_{3\mathtt{s}}\boldsymbol{s} &
\end{array} \quad . \qquad (1.6)$$

The type of the reflexive pronoun expresses the anaphoric content by first capturing the subject type with the right adjoint and then reproducing it twice, once in the form of the subject type itself and once in the form of the dummy-object type. The syntactic agreement is guaranteed by the fact that the indices for person, gender and number are identical in the three simple types. We find reductions for

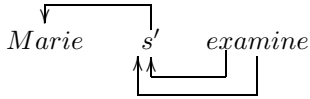
Marie *s'* *examine* — *Nous* *nous* *examinons*

(MARY HERSELF EXAMINES) — (WE OURSELVES EXAMINE)

$(\nu_{\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \pi_{3\mathtt{fs}}\ \hat{o}_{\mathtt{fs}})\ (\hat{o}^r\pi^r_{3\mathtt{s}}\ \boldsymbol{s})$ — $(\nu_{1\mathtt{fp}})\ (\pi^r_{1\mathtt{fp}}\ \pi_{1\mathtt{fp}}\ \hat{o}_{\mathtt{fp}})\ (\hat{o}^r\pi^r_{1\mathtt{p}}\ \boldsymbol{s})$,

but not for **Nous s'examinons*.

In the graph below, the two argument places are filled by the same entity to which the reflexive pronoun refers, namely *Marie*.



Here, the underlink under the subject type of *Marie* and its right adjoint in *s'* is represented by an arrow drawn above the words. The two under-arrows say that *s'* is both subject (=agent) and object (=patient) for `examiner`.

In fact, the translation algorithm into predicate logic identifies the subject with the dummy subject and the object. The translation of *Marie s'examine* is the atomic formula `examiner(marie,marie)`.

3.2. Modal verbs

The dependency is still captured, if the clitic is separated from the subject by modal verbs, like in *Marie doit vouloir s'examiner* (MARY MUST WANT TO EXAMINE HERSELF). To make the modal verb 'remember' the features of the subject, we add new basic types, the so-called *dummy subjects*

$$\hat{\pi}_{pgn}\,,\ p=1,2,3\,,g=\mathtt{m},\mathtt{f}\text{ and }n=\mathtt{s},\mathtt{p}$$

as well as a type $\hat{\pi}$ when the features do not matter. We postulate

$$\hat{\pi}_{pgn}\rightarrow\hat{\pi}\text{ for }p=1,2,3\,,g=\mathtt{m},\mathtt{f}\text{ and }n=\mathtt{s},\mathtt{p}\,.$$

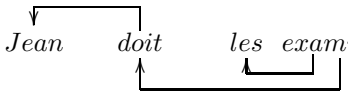
Besides the types for the modal verbs, we need two types for the infinitive, one for the reflexive and one for the non-reflexive case. We must also add a new type for the reflexive pronoun, when it is separated from the subject by a modal verb

$$\begin{array}{lcll}
\textit{doit} & : & \pi^r_{3g\mathtt{s}}\boldsymbol{s}\boldsymbol{i}^\ell\hat{\pi}_{3g\mathtt{s}} & \text{for } g=\mathtt{m},\mathtt{f}\\
\textit{vouloir} & : & \hat{\pi}^r_{pgn}\boldsymbol{i}\boldsymbol{i}^\ell\hat{\pi}_{pgn} & \text{for } p=1,2,3,\quad g=\mathtt{m},\mathtt{f},\quad n=\mathtt{s},\mathtt{p}\\
\textit{examiner} : & & \bar{o}^r\hat{\pi}^r\boldsymbol{i} & \\
\textit{examiner} : & & \hat{o}^r\hat{\pi}^r\boldsymbol{i} & \\
\textit{s'} & : & \hat{\pi}^r_{3gn}\hat{\pi}_{3gn}\hat{o}_{gn} & \text{for } g=\mathtt{m},\mathtt{f},\quad n=\mathtt{s},\mathtt{p}\\
\textit{s'} & : & \hat{\pi}^r_{3gn}\hat{\pi}_{3gn}\hat{\omega}_3 & .
\end{array}\tag{1.7}$$

We find the reductions to the sentence type

$$\begin{array}{l}
\textit{Jean}\qquad\textit{doit}\qquad\textit{les}\quad\textit{examiner}\\
(\underbrace{\nu_{\mathtt{ms}}})\,(\underbrace{\pi^r_{3\mathtt{ms}}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \underbrace{\hat{\pi}_{3\mathtt{ms}}})\,(\underbrace{o_{3g\mathtt{p}}})\,(\underbrace{\bar{o}^r\ \hat{\pi}^r\ \boldsymbol{i}})\,,\text{where } g=\mathtt{m},\mathtt{f}\,.
\end{array}$$

Note that the dummy-subject type in the infinitive of the transitive verb *examiner* expresses the dependency of the corresponding relation `examiner` on the agent `jean`



The long under-arrow corresponds to the underlink from the right-adjoint of the dummy subject in *examiner* to the dummy subject in *doit*. If we continue this arrow by the over-arrow from *doit* to *Jean*, we get the dependency arrow.

The dummy-subject also serves a syntactical purpose in modal verbs. It precludes **Tu dois s'examiner*, but accepts

Marie *doit* *s'* *examiner*
(MARY MUST HERSELF EXAMINE)
$(\underbrace{\nu_{\mathtt{fs}}})\ (\underbrace{\pi^r_{3\mathtt{fs}}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \underbrace{\hat{\pi}_{3\mathtt{fs}}})\,(\underbrace{\hat{\pi}^r_{3\mathtt{fs}}}\ \underbrace{\hat{\pi}_{3\mathtt{fs}}}\ \underbrace{\hat{o}_{\mathtt{fs}}})\,(\underbrace{\hat{o}^r}\ \hat{\pi}^r\ \boldsymbol{i})\,.$

and

Marie *doit* *vouloir* *s'* *examiner*
(MARY MUST WANT HERSELF EXAMINE)
$(\underbrace{\nu_{\mathtt{fs}}})\ (\underbrace{\pi^r_{3\mathtt{fs}}}\ \boldsymbol{s}\ \underbrace{\boldsymbol{i}^\ell\ \hat{\pi}_{3\mathtt{fs}}})\,(\underbrace{\hat{\pi}^r_{3\mathtt{fs}}}\ \boldsymbol{i}\ \boldsymbol{i}^\ell\ \underbrace{\hat{\pi}_{3\mathtt{fs}}})\ (\underbrace{\hat{\pi}^r_{3\mathtt{fs}}}\ \underbrace{\hat{\pi}_{3\mathtt{fs}}}\ \underbrace{\hat{o}_{\mathtt{fs}}})\,(\underbrace{\hat{o}^r}\ \hat{\pi}^r\ \boldsymbol{i})\,.$

3.3. Agreement with the past participle

In the compound past of the active form, the past participle agrees in gender and number with the direct object clitic. This is the reason why the basic types $\hat{o}_{gn}$ depend on these features. The clitics precede the auxiliary, which in its turn precedes the past participle. Transitive verbs form the compound past with the auxiliary *avoir* unless a reflexive clitic precedes the auxiliary. In the latter case the auxiliary is *être*. Some intransitive verbs form the compound past with *être*, others with *avoir*. The passive is always formed with *être*. If the verb is in passive mode or forms its compound past with the auxiliary *être*, the past participle agrees in gender and number with the subject.

We add the following types to the dictionary, covering both the postverbal object and the preverbal direct object clitic. We consider both non-reflexive and reflexive clitics.

examiné:	$\hat{\pi}^r\boldsymbol{p}o^\ell$	*examiné*	$: \hat{o}^r_{\mathtt{ms}}\hat{\pi}^r\boldsymbol{p}$	*avoir*:	$o^r_{pgn}\hat{\pi}^r\boldsymbol{i}\boldsymbol{p}^\ell\hat{\pi}\hat{o}_{gn}$
avoir	$: \hat{\pi}^r_{pgn}\boldsymbol{i}\boldsymbol{p}^\ell\hat{\pi}_{pgn}$	*examinée*	$: \hat{o}^r_{\mathtt{fs}}\hat{\pi}^r\boldsymbol{p}$	*a*	$: o^r_{pgn}\pi^r_{3\mathtt{s}}\boldsymbol{s}\boldsymbol{p}^\ell\hat{\pi}\hat{o}_{gn}$
a	$: \pi^r_{3\mathtt{gs}}\boldsymbol{s}\boldsymbol{p}^\ell\hat{\pi}_{3\mathtt{gs}}$	*examinés*	$: \hat{o}^r_{\mathtt{mp}}\hat{\pi}^r\boldsymbol{p}$	*est*	$: \hat{o}^r_{3\mathtt{gs}}\pi^r_{3\mathtt{s}}\boldsymbol{s}\boldsymbol{p}^\ell\hat{\pi}\hat{o}_{\mathtt{gs}}$
dû	$: \hat{\pi}^r_{pgn}\boldsymbol{p}\boldsymbol{i}^\ell\hat{\pi}_{pgn}$	*examinées*:	$\hat{o}^r_{\mathtt{fp}}\hat{\pi}^r\boldsymbol{p}$	*être*	$: \hat{o}^r_{pgn}\hat{\pi}^r\boldsymbol{i}\boldsymbol{p}^\ell\hat{\pi}\hat{o}_{gn}$

(1.8)

for $p = 1, 2, 3$, $g = \mathtt{m}, \mathtt{f}$, $n = \mathtt{s}, \mathtt{p}$.
The types of *avoir* and *être* differ only by their first simple type. It is o^r_{pgn} for the former and $\hat{o}^r_{pgn}$ for the latter. This precludes **Marie les est examinés*.

Jean *a* *examiné* *Marie*
(JOHN HAS EXAMINED MARY)
$(\nu_{\mathtt{ms}})\ (\pi^r_{3\mathtt{ms}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{\pi}_{3\mathtt{ms}})\ (\hat{\pi}^r\ \boldsymbol{p}\ o^\ell)\ (\nu_{\mathtt{fs}})$

Marie les a examinés
$(\nu_{\mathtt{fs}})\ (o_{3\mathtt{mp}})\ (o^r_{3\mathtt{mp}}\ \pi^r_{3\mathtt{s}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{\pi}\ \hat{o}_{\mathtt{mp}})\ (\hat{o}^r_{\mathtt{mp}}\ \hat{\pi}^r\ \boldsymbol{p})$

Marie s' est examinée
$(\nu_{\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \pi_{3\mathtt{fs}}\ \hat{o}_{3\mathtt{fs}})\ (\hat{o}^r_{3\mathtt{fs}}\ \pi^r_{3\mathtt{s}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{\pi}\ \hat{o}_{3\mathtt{fs}})\ (\hat{o}^r_{\mathtt{fs}}\ \hat{\pi}^r\ \boldsymbol{p})$.

Similarly we cannot derive the non-grammatical **Nous leurs sommes parlé* nor **Nous nous avons parlé*, but we find

Nous nous sommes parlé
$(\nu_{1\mathtt{fp}})\ (\pi^r_{1\mathtt{fp}}\ \pi_{1\mathtt{fp}}\ \hat{\omega}_1)\ (\hat{\omega}^r_1\ \pi^r_{1\mathtt{fp}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{\pi}\ \hat{\omega})\ (\hat{\omega}^r\ \hat{\pi}^r\ \boldsymbol{p})$.

We may also combine modal verbs with the past participle

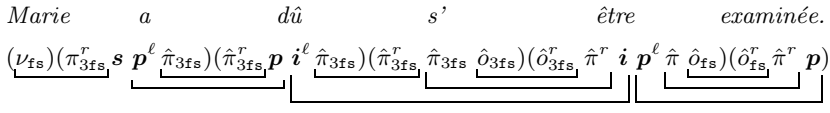
Marie doit les avoir examinés
$(\nu_{\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \hat{\pi}_{3\mathtt{fs}})\ (o_{3\mathtt{mp}})\ (o^r_{3\mathtt{mp}}\ \hat{\pi}^r\ \boldsymbol{i}\ \boldsymbol{p}^\ell\ \hat{\pi}\ \hat{o}_{\mathtt{mp}})\ (\hat{o}^r_{\mathtt{mp}}\ \hat{\pi}^r\ \boldsymbol{p})$.

The dummy subject type in the auxiliary *a , avoir* in the sentences above can only be justified semantically at this stage. It means that the transitive verb is viewed as a binary relation in all its tenses and modalities.

However, the role of the dummy subject in the past participle of modal verbs becomes obvious in sentences like *Marie a dû s'être examinée* in opposition to **Marie a dû t'être examinée*, **Marie a dû s'être examinés*. The dummy subject-type in the auxiliary transmits

the features of person, gender and number of the subject throughout the sentence. In Subsection 1, we show how French syntax uses the dummy subject in the infinitive for distinguishing between the agent (intended subject) and the patient (intended object). Hence dummy types serve a semantical and syntactical purpose similar to that of an index in HPSG's: it is used to express unbounded dependencies. The increase of the length of some types seems not a heavy price to pay. The pregroup grammar proposed here is still an ordinary pregroup grammar with its efficient computability properties.

The dummy subject type in the auxiliary *a* and in the past participle *dû* of the modal verb *devoir* guarantees correct syntax in sentences like



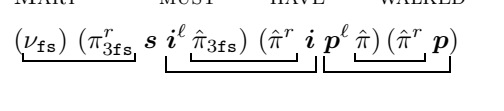
and even *Marie a dû vouloir s'être examinée* etc.

3.4. Intransitive verbs

Intransitive verbs form the compound past either with *avoir* or with *être*. In the former case, the past participle remains invariant. In the latter case, it agrees with the subject in gender and number. Hence, we add a new basic type $\boldsymbol{p}'$ to the dictionary for verbs forming their past participle with *être*. For example

$$\begin{array}{lcl}
a & : & \pi^r_{3\mathtt{s}}\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{\pi} \\
avoir & : & \hat{\pi}^r\boldsymbol{i}\boldsymbol{p}^{\ell}\hat{\pi} \\
marche & : & \pi^r_{3\mathtt{s}}\,\boldsymbol{s} \\
marcher & : & \hat{\pi}^r\,\boldsymbol{i} \\
marché & : & \hat{\pi}^r\boldsymbol{p}
\end{array}
\qquad
\begin{array}{lcll}
est & : & \pi^r_{3gs}\boldsymbol{s}\boldsymbol{p}'^{\ell}\hat{\pi}_{gs} & \text{for } g=\mathtt{m},\mathtt{f} \\
être & : & \hat{\pi}^r_{gn}\boldsymbol{i}\boldsymbol{p}'^{\ell}\hat{\pi}_{gn} & \text{for } g=\mathtt{m},\mathtt{f};\ n=\mathtt{s},\mathtt{p} \\
part & : & \pi^r_{3\mathtt{s}}\,\boldsymbol{s} & \\
partir & : & \hat{\pi}^r\,\boldsymbol{i} & \\
parti & : & \hat{\pi}^r_{\mathtt{ms}}\boldsymbol{p}' & \\
partie & : & \hat{\pi}^r_{\mathtt{fs}}\boldsymbol{p}' & \\
partis & : & \hat{\pi}^r_{\mathtt{mp}}\boldsymbol{p}' & \\
parties & : & \hat{\pi}^r_{\mathtt{fp}}\boldsymbol{p}' &
\end{array}
\tag{1.9}$$

We find the following reductions to the sentence type



and

$$\begin{array}{llll} \textit{Marie} & \textit{doit} & \textit{être} & \textit{partie} \\ \textsc{Mary} & \textsc{must} & \textsc{have} & \textsc{left} \\ (\underbrace{\nu_{\mathtt{fs}}}\ (\pi^r_{3\mathtt{fs}}\ \underbrace{\boldsymbol{s}\ \boldsymbol{i}^\ell\ \hat{\pi}_{3\mathtt{fs}}}\ (\hat{\pi}^r_{\mathtt{fs}}\ \underbrace{\boldsymbol{i}\ \boldsymbol{p}'^\ell\ \hat{\pi}_{\mathtt{fs}}}\ (\hat{\pi}^r_{\mathtt{fs}}\ \underbrace{\boldsymbol{p}'}) \end{array}.$$

Note that the new basic type $\boldsymbol{p}'$ prevents **Marie doit avoir partie* and **Marie doit être marché.*

3.5. Passive

For the passive form of transitive verbs yet more entries are needed. In particular, we introduce a new basic type $\tilde{\pi}$ for the agent of the passive form .

$$\begin{array}{ll} \textit{être} & : \hat{\pi}^r_{gn}\boldsymbol{i}\boldsymbol{p}^\ell\hat{o}_{gn} \\ \textit{est} & : \pi^r_{3gs}\boldsymbol{s}\boldsymbol{p}^\ell\hat{o}_{gs} \\ \textit{examiné} & : \hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\tilde{\pi}^\ell \\ \textit{examinée} & : \hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell \\ \textit{examinés} & : \hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\tilde{\pi}^\ell \\ \textit{examinées} & : \hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\tilde{\pi}^\ell \\ \textit{par} & : \tilde{\pi}o^\ell \end{array} \qquad \begin{array}{ll} \textit{examiné} & : \hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi} \\ \textit{examinée} & : \hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi} \\ \textit{examinés} & : \hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi} \\ \textit{examinées} & : \hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi} \end{array} \quad , \text{ for } g = \mathtt{m}, \mathtt{f}, n = \mathtt{s}, \mathtt{p}.$$

(1.10)

Choosing the entries $\textit{est} : \pi^r_{3\mathtt{fs}}s\hat{o}_{\mathtt{fs}}\boldsymbol{p}^\ell$ and $\textit{examinée} : \boldsymbol{p}\hat{o}^r_{\mathtt{fs}}\tilde{\pi}^\ell$, we find the following reduction

$$\begin{array}{lll} \textit{Marie} & \textit{est} & \textit{examinée par Jean} \\ \textsc{mary} & \textsc{is} & \textsc{examined by john} \\ \underbrace{(\nu_{\mathtt{fs}})(\pi^r_{3\mathtt{fs}},}\ \underbrace{\boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{o}_{\mathtt{fs}})(\hat{o}^r_{\mathtt{fs}},}\ \underbrace{\boldsymbol{p}\ \tilde{\pi}^\ell)(\tilde{\pi}\ o^\ell)(\nu_{\mathtt{ms}})} \end{array}.$$

Finally, if the agent of the passive is absent, like in *Marie est examinée*, we can express this by choosing the right-hand type above for the past participle. We find the following parsing

$$\begin{array}{lll} \textit{Marie} & \textit{est} & \textit{examinée} \\ (\textsc{mary} & \textsc{is} & \textsc{examined}) \\ (\underbrace{\nu_{\mathtt{fs}})}\ (\pi^r_{3\mathtt{fs}},\ \underbrace{\boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{o}_{\mathtt{fs}})}\ (\hat{o}^r_{\mathtt{fs}},\ \underbrace{\boldsymbol{p}\ \tilde{\pi}^\ell\tilde{\pi}}) \end{array}.$$

3.6. Causal verbs

Verbs like *faire, laisser, entendre, laisser* take infinitival verb phrases as complements. The syntactical agreement of the intended subject-complement of the infinitive may be different from that of the intended object-complement of the infinitive. As an example we consider the verb *entendre*. It is transitive and as such has the same types as *estimer*. It also takes an infinitival phrase as complement. The dictionary therefore has the following additional entries

$$\begin{array}{ll} \textit{entend} : \pi^r_{3\mathtt{s}}\, \boldsymbol{s}\, \boldsymbol{i}^\ell \\ \textit{entend} : \pi^r_{3\mathtt{s}}\, \boldsymbol{s}\, \boldsymbol{i}^\ell \hat{\pi} o^\ell \\ \textit{entend} : \bar{o}^r \pi^r_{3\mathtt{s}}\, \boldsymbol{s}\, \boldsymbol{i}^\ell \hat{o} \\ \textit{entend} : \bar{o}^r \pi^r_{3\mathtt{s}}\, \boldsymbol{s}\, \boldsymbol{i}^\ell \hat{\pi} \end{array} \qquad \begin{array}{ll} \textit{jouer} : \hat{\pi}^r \boldsymbol{i}\, o^\ell \\ \textit{jouer} : \boldsymbol{i}\, \hat{\pi} \hat{\pi}^r \bar{o} \bar{o}^r \\ \textit{jouer} : \boldsymbol{i}\, \hat{\pi} \hat{\pi}^r o^\ell \\ \textit{jouer} : \hat{\pi}^r \boldsymbol{i}\, \bar{o} \bar{o}^r \\ \textit{jouer} : \hat{o}^r \boldsymbol{i}\, \hat{\pi} \hat{\pi}^r \end{array} \qquad \begin{array}{ll} \textit{entendu} & : \hat{\pi}^r \boldsymbol{p}\, \boldsymbol{i}^\ell \\ \textit{entendu} & : \hat{\pi}^r \boldsymbol{p}\, \boldsymbol{i}^\ell \hat{\pi} o^\ell \\ \textit{entendu} & : \hat{o}^r \hat{\pi}^r \boldsymbol{p} \boldsymbol{i}^\ell \hat{o} \\ \textit{entendu} & : \hat{o}^r_{\mathtt{ms}} \hat{\pi}^r \boldsymbol{p} \boldsymbol{i}^\ell \hat{\pi} \\ \textit{entendue} & : \hat{o}^r_{\mathtt{fs}} \hat{\pi}^r \boldsymbol{p} \boldsymbol{i}^\ell \hat{\pi} \\ \textit{entendus} & : \hat{o}^r_{\mathtt{mp}} \hat{\pi}^r \boldsymbol{p} \boldsymbol{i}^\ell \hat{\pi} \\ \textit{entendues} & : \hat{o}^r_{\mathtt{fp}} \hat{\pi}^r \boldsymbol{p} \boldsymbol{i}^\ell \hat{\pi} \end{array}$$

The speaker can mention both the subject and the object of the infinitive, none of them or one or the other.

Neither subject nor object of the infinitive mentioned

$$\begin{array}{l} \textit{Marie} \quad \textit{entend} \quad \textit{jouer} \\ (\textsc{mary} \quad \textsc{hears} \quad \textsc{play}) \\ (\underbrace{\nu_{\mathtt{fs}}})\ (\pi^r_{3\mathtt{s}}\ \underbrace{\boldsymbol{s}\ \boldsymbol{i}^\ell})\ (\underbrace{\boldsymbol{i} \hat{\pi} \hat{\pi}^r}\ \underbrace{\bar{o} \bar{o}^r})\,. \end{array}$$

Subject of infinitive mentioned

$$\begin{array}{l} \textit{Marie} \quad \textit{entend} \quad \textit{des} \quad \textit{musiciens} \quad \textit{jouer} \\ (\textsc{mary} \quad \textsc{hears} \quad \textsc{some} \quad \textsc{musicians} \quad \textsc{play}) \\ (\underbrace{\nu_{\mathtt{fs}}})\ (\pi^r_{3\mathtt{s}}\ \boldsymbol{s}\ \underbrace{\boldsymbol{i}^\ell \hat{\pi} o^\ell})\ (\boldsymbol{n}_{\mathtt{mp}} \underbrace{c^\ell_{\mathtt{mp}}})\ (c_{\mathtt{mp}})(\hat{\pi}^r\ \boldsymbol{i}\ \underbrace{\bar{o} \bar{o}^r})\,. \end{array}$$

Object of infinitive mentioned

$$\begin{array}{l} \textit{Marie} \quad \textit{entend} \quad \textit{jouer} \quad \textit{des} \quad \textit{valses} \\ (\textsc{mary} \quad \textsc{hears} \quad \textsc{play} \quad \textsc{some waltzes}) \\ (\underbrace{\nu_{\mathtt{fs}}})\ (\pi^r_{3\mathtt{s}}\ \underbrace{\boldsymbol{s}\ \boldsymbol{i}^\ell})\ (\underbrace{\boldsymbol{i} \hat{\pi} \hat{\pi}^r} \underbrace{o^\ell})\ (\boldsymbol{n}_{\mathtt{fp}} \underbrace{c^\ell_{\mathtt{fp}}})\ (c_{\mathtt{fp}})\,. \end{array}$$

With the preverbal object clitic, this syntactical difference disappears in the simple tenses, but there are still two semantical interpretations, captured by the two reductions

Subject of infinitive: *les* = THE MUSICIANS

$$\begin{array}{l} \textit{Marie} \quad \textit{les} \quad \textit{entend} \quad \textit{jouer} \\ (\text{MARY THEM} \quad \text{HEARS} \quad \text{PLAY}) \\ (\nu_{\mathtt{fs}})\ (o_{3gp})(\bar{o}^r\,\pi^r_{3\mathtt{s}})\ \boldsymbol{s}\ \boldsymbol{i}^{\ell}\,\hat{\pi})\,(\,\hat{\pi}^r\ \boldsymbol{i}\ \bar{o}\bar{o}^r)\,. \end{array}$$

Object of infinitive: *les* = THE WALTZES

$$\begin{array}{l} \textit{Marie} \quad \textit{les} \quad \textit{entend} \quad \textit{jouer} \\ (\text{MARY THEM} \quad \text{HEARS} \quad \text{PLAY}) \\ (\nu_{\mathtt{fs}})\ (o_{3gp})(\bar{o}^r\,\pi^r_{3\mathtt{s}})\ \boldsymbol{s}\ \boldsymbol{i}^{\ell}\,\hat{o})\,(\,\hat{o}^r\ \boldsymbol{i}\ \hat{\pi}\hat{\pi}^r)\,. \end{array}$$

In the compound tenses the semantical difference is again expressed by a different syntax. The past participle of the causal verb agrees with the clitic if it is the intended subject of the infinitive, but remains invariant if the clitic is the intended object.

Subject of infinitive: *les* = THE MUSICIANS

$$\begin{array}{l} \textit{Marie} \quad \textit{les} \quad \textit{a} \quad \textit{entendus} \quad \textit{jouer} \\ (\text{MARY THEM} \quad \text{HAS} \quad \text{HEARD} \quad \text{PLAY}) \\ (\nu_{\mathtt{fs}})\ (o_{3\mathtt{mp}})(o_{3\mathtt{mp}}{}^r\,\pi^r_{3\mathtt{s}})\ \boldsymbol{s}\,\boldsymbol{p}^{\ell}\,\hat{\pi}\,\hat{o}_{\mathtt{mp}})\ (\hat{o}^r_{\mathtt{mp}}\,\hat{\pi}^r\ \boldsymbol{p}\ \boldsymbol{i}^{\ell}\,\hat{\pi})\ (\hat{\pi}^r\ \boldsymbol{i}\,\bar{o}^r\bar{o})\,. \end{array}$$

Object of infinitive: *les* = THE WALTZES

$$\begin{array}{l} \textit{Marie} \quad \textit{les} \quad \textit{a} \quad \textit{entendu} \quad \textit{jouer} \\ (\text{MARY THEM} \quad \text{HAS} \quad \text{HEARD} \quad \text{PLAY}) \\ (\nu_{\mathtt{fs}})\ (o_{3\mathtt{fp}})(o_{3\mathtt{fp}}{}^r\,\pi^r_{3\mathtt{s}})\ \boldsymbol{s}\,\boldsymbol{p}^{\ell}\,\hat{\pi}\,\hat{o}_{\mathtt{fp}})\ (\hat{o}^r\,\hat{\pi}^r\ \boldsymbol{p}\ \boldsymbol{i}^{\ell}\,\hat{o})\ (\hat{o}^r\,\boldsymbol{i}\,\hat{\pi}\hat{\pi}^r)\,. \end{array}$$

Note that the causal verb captures the semantic and syntactic differences, by either introducing a dummy subject or a dummy object.

4. Linear parsing

The reader who just has looked at the preceding section may wonder if the explosion of types does not increase computational complexity. We we'll show that this is not the case. On the contrary, the preceding grammar and others, similar ones, can be parsed by a linear algorithm.

First, any pregroup grammar can be recognized by an algorithm running in time proportional to n^3, where n is the length of the word, see for example [7] or [5]. Moreover, the constant factor of n^3 is a multiple of k^3p^3, where k is a bound to the number of types per word and p a bound to the length of the type in the dictionary. So, only the constant factor increases with the length of the types or the number of types per word. This cubic algorithm interweaves type assignment and type checking. Hence, ambiguity of kind I, due to the existence of several type assignments, and ambiguity of kind II, due to multiple reductions per type assignment, are also intermingled.

To show that the complexity can actually be lowered to linear time, we separate the two ambiguity factors. Our first step is to construct an algorithm, which for every type assignment $T_1, \ldots, T_n$ constructs a reduction of the concatenated type $T_1 \ldots T_n$ to an irreducible form. If this irreducible form is the sentence type, the algorithm has constructed a parsing of the sentence. The reduction of $T_1 \ldots T_n$ is constructed in time proportional to the length of $T_1 \ldots T_n$. The algorithm is complete if the dictionary has no critical triples like the dictionary of the preceding section.

The second step is to define conditions on the dictionary so that the number of non-losing type assignments is bounded by a constant. The structure of the types in the dictionary gives us a criterion to recognize type assignments without a reduction to the sentence type, and this after only an initial segment of the type assignment has been processed. Hence the algorithm does not process any extension of such an initial segment. Moreover, if the different types of a word are similar, we compute a reduction only once.

4.1. Choosing type assignments

We start with an example how the algorithm defined in the next subsection finds a type assignment and a corresponding reduction.

$$\begin{array}{cccc} \textit{Marie} & \textit{doit} & \hat{e}\textit{tre} & \textit{examin\'ee} \\ (\underbrace{\nu_{\mathtt{fs}}})\ (\underbrace{\pi^r_{3\mathtt{fs}}}\boldsymbol{s}\,\boldsymbol{i}^{\ell}\,\underbrace{\hat{\pi}_{3\mathtt{fs}}})\ (\underbrace{\hat{\pi}^r_{\mathtt{fs}}}\boldsymbol{i}\,\boldsymbol{p}^{\ell}\,\underbrace{\hat{o}_{\mathtt{fs}}})\ (\underbrace{\hat{o}^r_{\mathtt{fs}}}\boldsymbol{p}\,\underbrace{\tilde{\pi}^{\ell}\tilde{\pi}_1}) \end{array}.$$

The algorithm processes the string of words as 'we hear them', i.e. from left to right. It keeps two memories, the S-memory stack where the irreducible form of the processed string is stored, and the$R$-memory for the reduction computed so far. For each word, it considers all its types in the dictionary. Each type is read from left to right. The simple type being read is compared with the last type of the irreducible form in the memory. If the two contract, the link is added to the reduction memory and the last type is popped. If not, the simple type being read is added on top of the stack. If no extension of the updated irreducible form can be reduced to the sentence type, the algorithm declares the type assignment processed so far a a 'loser'. No extension of a 'loser' will be processed by the algorithm.

The criterion used here for recognizing losers is that the type a^z being read does not contract with the last type in the memory and that for no basic type b in the same component as a, the dictionary has an occurrence of a^{z+1} .

1) *Marie*:
Store $T_1 = \nu_{\mathtt{fs}}$ in the S_{T_1}-memory

$$\begin{array}{c} \textit{Marie} \\ (\nu_{\mathtt{fs}}) \end{array}$$

2) $\textit{doit} : \pi^r_{3g\mathtt{s}}\boldsymbol{s}\boldsymbol{i}^{\ell}\hat{\pi}_{3g\mathtt{s}}$ for $g = \mathtt{m}, \mathtt{f}$
The type assignment $(\nu_{\mathtt{fs}})(\pi^r_{3\mathtt{ms}}\boldsymbol{s}\boldsymbol{i}^{\ell}\hat{\pi}_{3\mathtt{ms}})$ is a loser, because $\nu_{\mathtt{fs}}\hat{\pi}^r_{3\mathtt{ms}} \not\to 1$. The type $\hat{\pi}^r_{3\mathtt{ms}}$ cannot be canceled from the right, because there are no double right adjoints in the dictionary. Hence the irreducible string $\nu_{\mathtt{fs}}\hat{\pi}^r_{3\mathtt{ms}}$ will remain in S for all extensions. Here losers are identified when the first type of a possible type assignment for the processed word is read. Though this happens quite frequently, in general the considered type has to processed till the end before it is eliminated or accepted as a reasonable type assignment for the word.

Choose $T_2 = \pi^r_{3\mathtt{fs}}\boldsymbol{s}\boldsymbol{i}^{\ell}\hat{\pi}_{3\mathtt{fs}}$.

Store the link $\{1,2\}$ under $\nu_{\mathtt{fs}}\pi^r_{3\mathtt{fs}}$ in the $R_{T_1T_2}$-memory. Pop $\nu_{\mathtt{fs}}$ from the $S_{T_1T_2}$-memory.

Store successively $\boldsymbol{s}$, $\boldsymbol{i}^\ell$ and $\hat{\pi}_{3\mathtt{fs}}$ on top of the S-memory stack.

$$\begin{array}{ll} \textit{Marie} & \textit{doit} \\ (\underbrace{\nu_{\mathtt{fs}}}) & (\underbrace{\pi^r_{3\mathtt{fs}}}\boldsymbol{s}\boldsymbol{i}^\ell\hat{\pi}_{3\mathtt{fs}}) \end{array}$$

3)

$$\textit{être} \quad : \quad \begin{array}{l} \hat{o}^r_{gn}\pi^r\boldsymbol{i}\boldsymbol{p}^\ell\hat{\pi}\hat{o}_{gn} \\ \hat{\pi}^r_{gn}\boldsymbol{i}\boldsymbol{p}'^\ell\hat{\pi}_{gn} \\ \hat{\pi}^r_{gn}\boldsymbol{i}\boldsymbol{p}^\ell\hat{o}_{gn} \end{array} \qquad \text{for } g=\mathtt{m},\mathtt{f}\,;\ n=\mathtt{s},\mathtt{p}.$$

The types starting with $\hat{o}^r_{gn}$ are losers

The types starting with $\hat{\pi}^r_{gn}$ are a losers if $g=\mathtt{m}$ or $n=\mathtt{p}$.

Choose $T_3=\hat{\pi}^r_{\mathtt{fs}}\boldsymbol{i}\boldsymbol{p}^\ell\hat{o}_{\mathtt{fs}}$

Store the link under $\hat{\pi}_{\mathtt{fs}}\hat{\pi}^r_{\mathtt{fs}}$ in the R-memory. Pop $\hat{\pi}_{\mathtt{fs}}$ from the S-memory.

Store the link under $\boldsymbol{i}^\ell\boldsymbol{i}$ in the R-memory. Pop $\boldsymbol{i}^\ell$ from the S-memory.

Store successively $\boldsymbol{p}^\ell$ and $\hat{o}_{\mathtt{fs}}$.

The irreducible form in $S_{T_1T_2T_3}$ is $\boldsymbol{s}\boldsymbol{p}^\ell\hat{o}_{\mathtt{fs}}$.

$$\begin{array}{lll} \textit{Marie} & \textit{doit} & \textit{être} \\ (\underbrace{\nu_{\mathtt{fs}}}) & (\underbrace{\pi^r_{3\mathtt{fs}}}\boldsymbol{s}\,\underbrace{\boldsymbol{i}^\ell\,\hat{\pi}_{\mathtt{fs}}}) & (\underbrace{\hat{\pi}^r_{\mathtt{fs}}}\boldsymbol{i}\boldsymbol{p}^\ell\hat{o}_{\mathtt{fs}}) \end{array}$$

Choose $T'_3=\hat{\pi}^r_{\mathtt{fs}}\boldsymbol{i}\boldsymbol{p}'^\ell\hat{\pi}_{\mathtt{fs}}$

The irreducible form in $S_{T_1T_2T'_3}$ now is $\boldsymbol{s}\boldsymbol{p}'^\ell\hat{\pi}_{\mathtt{fs}}$.

$$\begin{array}{lll} \textit{Marie} & \textit{doit} & \textit{être} \\ (\underbrace{\nu_{\mathtt{fs}}}) & (\underbrace{\pi^r_{3\mathtt{fs}}}\boldsymbol{s}\,\underbrace{\boldsymbol{i}^\ell\,\hat{\pi}_{3\mathtt{fs}}}) & (\underbrace{\hat{\pi}^r_{\mathtt{fs}}}\boldsymbol{i}\boldsymbol{p}'^\ell\hat{\pi}_{\mathtt{fs}}) \end{array}$$

4)

$$\textit{examinée} \quad : \quad \begin{array}{l} \hat{o}^r_{\mathtt{fs}}\hat{\pi}^r\boldsymbol{p} \\ \hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell \\ \hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi} \end{array} \quad .$$

The three types are losers when chosen after $S_{T_1T_2T'_3}$. Extend $T_1T_2T_3$. Starting with $S_{T_1T_2T_3}$ and $R_{T_1T_2T_3}$, only the three types

$T_4=\hat{o}^r_{\mathtt{fs}}\hat{\pi}^r\boldsymbol{p}$

$T'_4=\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell$

$T_4'' = \hat{o}^r_{\mathtt{fs}} \boldsymbol{p} \tilde{\pi}^\ell \tilde{\pi}$
are processed beyond the first simple type. The first, T_4 is recognized as a loser when its second simple type is read, T_4' is processed at to the end and recognized as loser at this step. Finally for $T_1T_2T_3T_4''$ the following reduction is computed.

$$\begin{array}{llllll} \textit{Marie} & \textit{doit} & & \textit{être} & & \textit{examinée} \\ (\underbrace{\nu_{\mathtt{fs}}}) & (\underbrace{\pi^r_{3\mathtt{fs}}} & \boldsymbol{s} & \boldsymbol{i}^\ell \underbrace{\hat{\pi}_{3\mathtt{fs}}}) & (\underbrace{\hat{\pi}^r_{\mathtt{fs}}} & \boldsymbol{i} \; \boldsymbol{p}^\ell \underbrace{\hat{o}_{\mathtt{fs}}}) \; (\underbrace{\hat{o}^r_{\mathtt{fs}} \boldsymbol{p}} \; \underbrace{\tilde{\pi}^\ell \tilde{\pi}}) \, . \end{array}$$

The algorithm not only finds a reduction to the sentence type but also that it is the only one.

4.2. Parsing Algorithm

We define the algorithm above formally and prove the relevant properties. It is an adaptation of the linear processing algorithm in [11]. By Fact 1 of Section 2, it suffices that the algorithm searches for and produces reductions to the empty string.

A string of words $v_1 \dots v_n$ defines a set of *stages* consisting of triples
$s = (l, T_1 \dots T_l, p)$ where
l is the number of the word v_l being processed
$T_k = t_{k1} \dots t_{kq_k} \in D(v_k),\ 1 \leq k \leq l$,
p is a *position* in T_l, $0 \leq p \leq q_l$.
The stages are partially ordered as follows

$$(l, T_1 \dots T_l, p) \leq (l', T_1' \dots T_{l'}', p') \Leftrightarrow l \leq l',\ p \leq p',\ T_k = T_k' \text{ for } 1 \leq k \leq l \, .$$

To these we add an initial stage s_{in} such that $s_{in} < s$ for all s.

We remark that every stage s except the initial one has a unique immediate predecessor, which we denote by $s - 1$, i.e.

$$(l, T_1 \dots T_l, p) - 1 = \begin{cases} (l, T_1 \dots T_l, p-1), & \text{if } 1 \leq p\,; \\ (l-1, T_1 \dots T_{l-1}, q_{l-1}) & \text{if } p = 0 \text{ and } l > 1, \\ s_{in}, & \text{if } p = 0 \text{ and } l = 1. \end{cases}$$

The definitions imply that the set of stages smaller than or equal to a given stage s is totally ordered.

This total order can be used to control the way how the algorithm moves through the stages and to define the actual position $p(s)$ and

the type read at this position $t_{p(s)}$. At the initial stage $p(s_{in}) = 0$, $t_{p(s_{in})} = 1$. A stage of the form $(l, T_1 \dots T_l, 0)$, $1 \leq l \leq n$, is called a *downloading stage* and serves to choose a type $T_l \in D(v_l)$ for the word v_l. At a downloading stage $s = (l, T_1 \dots T_l, 0)$, the examined position remains unchanged

$$p(s) = p(s-1) = q_1 + \dots + q_{l-1} + 0\,.$$

After downloading, the string of simple types T_l is read from left to right. Each stage which is not initial and not downloading is called a *testing stage*. To reach the testing stage $s = (l, T_1 \dots T_l, p)$, $p \geq 1$, the preceding position $p(s-1)$ is incremented by 1:

$$p(s) = p(s-1) + 1 = q_1 + \dots + q_{l-1} + p\,.$$

It follows that the simple type occupying this position satisfies

$$t_{p(s)} = t_{lp}\,.$$

More generally, for every r such that $1 \leq r \leq p(s)$ there are a unique k and a unique p' such that $1 \leq k \leq l$, $1 \leq p' \leq q_k$ and $r = q_1 + \dots + q_{k-1} + p'$. We let

$$t_r = t_{kp'}\,.$$

The simple type $t_{p(s)}$ is tested for generalized contraction with the last not contracted type in the string. This test can be done in one time unit by accessing the partial order relation on the set of basic types. If it fails, $p(s)$ is added on the top of the stack indicating that $t_{p(s)}$ is the latest not (yet) contracted type. The other data remain unchanged. If the test succeeds, the stack is popped and the link consisting of the contracting positions is added to the reduction computed so far. As the test is only performed for non-initial and non-downloading stages, all positions r stored in the stack correspond to a unique number k and a unique p' for which $1 \leq p' \leq q_k$ and $r = q_1 + \dots + q_k + p'$.

Definition 2. ***Parsing Algorithm***
▼ *At the initial stage, let*

$$S(s_{in}) = \langle \varnothing, 0 \rangle\,, R(s_{in}) = \emptyset$$

▼ *At a downloading stage* $s = (l, T_1 \dots T_l, 0)$*, the stack and reduction remain unchanged*

$$S(s) = S(s-1),\ R(s) = R(s-1)$$

▼ *If $s(l, T_1 \dots T_l, p)$ is not downloading and not initial, let $top(s-1) = top(S(s-1))$. Then*

$$S(s) = \begin{cases} pop(S(s-1)), & \text{if } t_{top(s-1)}t_{p(s)} \to 1 \\ \langle S(s-1), p(s)\rangle, & \text{else} \end{cases}$$

$$R(s) = \begin{cases} R(s-1) \cup \{\{top(s-1), p(s)\}\}, & \text{if } t_{top(s-1)}t_{p(s)} \to 1 \\ R(s-1), & \text{else} \end{cases}.$$

For every stage s, define the string of simple types $T(s)$ inductively by

$$T(s_{in}) = t_0 = 1 \quad T(s) = T(s-1)t_{p(s)}.$$

It follows immediately from the definitions that $T(s)$ is the string of simple types processed up to stage s, i.e.

$$T(s) = T_1 \dots T_{l-1}t_{l1} \dots t_{lp}, \text{ for every stage } s = (l, T_1 \dots T_l, p).$$

Note that the algorithm only stores positions in the stack $S(s)$, but it is straight forward to show that these positions form a strictly increasing substring of $\{1, \dots, p(s)\}$ such that the top of the stack is the largest position in the stack. The substring $\overline{S(s)}$ of $T(s)$ corresponding to the positions stored in $S(s)$ is defined by induction on the stack

$$\overline{\langle \emptyset, 0\rangle} = t_0 \text{ and } \overline{\langle S', p'\rangle} = \overline{S'}\, t_{p'} \text{ for } S(s) = \langle S', p'\rangle$$

Lemma 1. *For every stage $s = (l, T_1 \dots T_l, p)$, the string $\overline{S(s)}$ is an irreducible substring of $T(s)$ and $R(s)$ is a reduction from $T(s)$ to $\overline{S(s)}$.*

Proof: The restriction of R and S to the set of stages less or equal to $s = (l, T_1 \dots T_l, p)$ is a particular case of the type checking algorithm in [10], applied to the type assignment $T_1 \dots T_l$. The property follows now from Theorem 6.5 in [*loc. cit.*].

Let $T_1 \dots T_n$ be a type assignment of $v_1 \dots v_n$ and $s = (n, T_1 \dots T_n, q_n)$ a final stage. According to the lemma above, $R(s)$ is a reduction to an irreducible form of $T_1 \dots T_n$. If this irreducible form happens to be the sentence type, the algorithm gives a parsing of this sentence. If this irreducible form is not the sentence type, however, we cannot conclude in general that $T_1 \dots T_n$ has no reductions to the sentence type. Hence, the algorithm is not complete unless we impose sufficient conditions on the dictionary. One of them is *linearity*:

Definition 3. Linearity
A critical triple *of a string of simple types* $t_1 \dots t_q$ *is a substring* $t_i \dots t_j \dots t_k$ *such that* $i < j < k$ *and*

$$t_i t_j \to 1,\ t_j t_k \to 1,$$
$$t_{i+1} \dots t_{j-1} \to 1,\ t_{j+1} \dots t_{k-1} \to 1 \ .$$

A string of simple types without critical triples is called linear*. A dictionary is* linear*, if all type assignments with some reduction to the sentence type are linear.*

Here is a condition for linearity which is easy to verify: If $a^{(u)}, b^{(v)}, c^{(w)}$ appear in the dictionary and the basic types a, b, c belong to the same connected component then $u = z-1, v = z, w = z+1$ is false for every integer z. The mini-dictionary of the preceding section satisfies this condition. For example the connected component of $\bar{o}$ has only basic types and right adjoints in the dictionary. The connected component of o only basic types and left adjoints.

The following two lemmas have been proved in [10].

Lemma 2 (Uniqueness of Reductions of Linear Strings)**.** *Every linear string of simple types* $t_1 \dots t_m$ *has a unique irreducible form* $t_{i_1} \dots t_{i_p}$ *and there is a unique reduction* R *such that* $R : t_1 \dots t_m \Rightarrow t_{i_1} \dots t_{i_p}$*.*

From this lemma follows at once that the algorithm defined above is complete for linear dictionaries.

Lemma 3 (**Completeness**)**.** *A string of words from a linear dictionary* $v_1 \dots v_n$ *is a sentence if and only if at some final stage* $s = (n, T_1 \dots T_n, q_n)$*, the reduction* $R(s)$ *reduces* $(T_1 \dots T_n)$ *to the sentence type.*

The algorithm is complete but inefficient as long as it computes all final stages. However, many intermediary stages cannot be extended to one with a reduction to the sentence type and therefore the Parsing Algorithm need not go through them. Hence we formulate a criterion which makes it possible to recognize such stages while running the Parsing Algorithm.

Definition 4. (Losing stages)
A simple type t *is* right cancellable *in* D*, if it is the sentence type* $\boldsymbol{s}$ *or if there is a simple type* t' *such that* $t \to t'$ *and* t'^r *occurs in* D*. A stage* $s = (l+1, T_1 \dots T_{l+1}, q_{l+1})$ *associated to a string of words* $v_1 \dots v_{l+1}$*,* $1 \le l < n$*, is* losing *if for some position* i *stored in* $S(s)$*, the simple type* t_i *is not right cancellable.*

Lemma 4 (Control)**.** *Assume that the dictionary is linear and let* $top(s)$ *denote the top of the stack* $S(s)$*. If at stage* s *the simple type* $t_{top(s)}$ *is not right cancellable, then no final stage extending* s *produces a reduction to the sentence type.*

Proof: Recall that $top(s)$ is the last element in the stack $S(s)$ and that the irreducible type defined by the stack ends with $t_{top(s)}$. The assumption then implies that $top(s)$ will never be popped from the stack. Therefore, for every extension s' of s, the irreducible form $\overline{S(s')}$ has an occurrence of $t_{top(s)} \neq \boldsymbol{s}$. Hence $R(s')$ is not a reduction to the sentence type. As the dictionary is linear, the type assignment defined by s' has no other irreducible forms.

Definition 5. (Controlled Parsing)
The Controlled Parsing Algorithm is defined as the Parsing Algorithm above except that it updates a predicate $losing(s)$ *to be true if* $t_{top(S)}$ *is not right cancellable. It moves to the down-loading stages* $s = (l, T_1 \dots T_l, 0)$*, for* $T_l \in D(v_l)$ *only if* $s = (l-1, T_1 \dots T_{l-1}, q_{l-1})$ *is not losing.*

Note that the existence of a unique reduction per type assignment does not preclude ambiguity, because several type assignments may have a reduction to the sentence type. However, we can formulate a condition on pregroup dictionaries so that the Controlled Parsing Algorithm only goes through a bounded number of stages when processing a word.

Definition 6. (Concise Dictionaries)
A dictionary is concise*, if for every sentence* $v_1 \dots v_n$ *and for every* $T_{l+1} \in D(v_{l+1})$*,* $1 \leq l < n$*, there is at most one stage* $(l, T_1 \dots T_l, q_l)$ *for which* $(l+1, T_1 \dots T_l T_{l+1}, q_{l+1})$ *is not losing.*

The sample dictionary of the previous section is concise. Conciseness makes the number of stages which have to be examined by the Parsing Algorithm proportional to the number of words.

Lemma 5 (**Bound on type assignments**)**.** *Let* D *be a concise dictionary and* k_0 *bound both the number of types per word and the length of types in the dictionary. Then for every string of words* $v_1 \dots v_n$ *and all* l*,* $1 \leq l \leq n$*, the number of non-losing stages* $(l, T_1 \dots T_l, q_l)$ *is bounded by* k_0*. Moreover, the number of basic steps performed by the Amended Parsing Algorithm while processing word* v_l *is constant.*

Proof: By induction on l. The property is obvious for $l = 1$, because there are at most k_0 lexical entries $v_l : T_1$ and the number of computa-

tion steps for every simple type of T_1 is bounded by a constant. Without loss of generality we may assume this constant is again k_0. When processing word v_{l+1}, only the non-losing stages $(l, T_1 \dots T_l, q_l)$ are extended by a lexical entry $T_{l+1} \in D(v_{l+1})$. By induction hypothesis, there are at most k_0 of non-losing stages. Moreover, there are at most k_0 types $T_{l+1} \in D(v_{l+1})$. Hence at most k_0^2 stages $(l+1, T_1 \dots T_l T_{l+1}, 0)$ are to be considered. As the length of T_{l+1} is bounded by k_0, the stage $(l+1, T_1 \dots T_l T_{l+1}, q_{l+1})$ is reached computing at most ${k_0}^2$ basic steps. As the dictionary is concise, at most one of the non-losing stages $(l, T_1 \dots T_l, q_l)$ will yield a non-losing stage $(l+1, T_1 \dots T_l T_{l+1}, q_{l+1})$ and this for every $T_{l+1} \in D(v_{l+1})$. Hence after performing at most k_0^4 basic steps for word v_{l+1}, the algorithm has processed all extensions of the non-losing type assignments $T_1 \dots T_l$ to v_{l+1} and recognized at most k_0 stages among them as non-losing.

Theorem 1 (**Linearity**). *For linear and concise pregroup grammars there is a complete linear algorithm which decides if $v_1 \dots v_n$ is a sentence and, if this is true, computes all parsings to the sentence type.*

Proof: The Parsing Algorithm considers all type assignments of a given string of words. As the dictionary has no critical triples, the Parsing Algorithm computes all reductions to the sentence type. The Controlled Parsing Algorithm skips only type assignments without a reduction to the sentence type. So it is still complete. Finally, if the grammar is concise, the run-time for each word is bounded by a constant.

5. Conclusion

The contribution of this paper is to show that an explosion of basic types does not necessarily increase run-time but may even diminish it. Similarly, increasing the number of types per word does not increase the complexity of the algorithm, but only the constant factor to which run-time is proportional. This constant can be lowered by exploiting certain regularities of features which make it possible to construct several distinct reductions by one and the same computation.

We have chosen the agreement with clitics in French because its syntactic structure reflects semantical complement control in the sense of [8]. The presented grammar puts the unexpressed subject or object in the right place, without any additional constructs like indexing. As we

stay within standard pregroup grammars, none of the computational efficiency is lost and, as we have shown, sometimes even improved

However, the fragment presented belongs to a class of pregroup grammars which are not expressive enough. Larger and more expressive language fragments are handled by non-linear dictionaries that also can be parsed by a linear algorithm for a given type assignment, see [12]. Future work will (attempt to) show that the conciseness criterion also can be used in this case and with the same results. Another line of investigation is to design similar pregroup grammars in other languages following, for example, the study of Germanic and Romance clitics in [3] or that of English verbs exhibiting uniform control constraints in [8].

6. Appendix

$$
\begin{array}{lcl}
par(\textsc{by}) & : & \tilde{\pi} o^{\ell} \\
Marie & : & \nu_{\mathtt{fs}} \\
Jean & : & \nu_{\mathtt{ms}}
\end{array}
$$

Clitics

$$
\begin{array}{lcl}
les & : & o_{3g\mathtt{p}} \\
leurs & : & \omega_{3g\mathtt{p}} \\
nous & : & \nu_{1g\mathtt{p}} \\
nous & : & \pi^r_{1g\mathtt{p}} \pi_{1g\mathtt{p}} \hat{o}_{1g\mathtt{p}} \\
nous & : & \hat{\pi}^r_{1g\mathtt{p}} \hat{\pi}_{1g\mathtt{p}} \hat{o}_{1g\mathtt{p}} \\
nous & : & \pi^r_{1g\mathtt{p}} \pi_{1g\mathtt{p}} \hat{\omega}_{1\mathtt{p}} \\
nous & : & \hat{\pi}^r_{1g\mathtt{p}} \hat{\pi}_{1g\mathtt{p}} \hat{\omega}_{1\mathtt{p}} \\
s' & : & \pi^r_{3gn} \pi_{3gn} \hat{o}_{3gn} \\
s' & : & \hat{\pi}^r_{3gn} \hat{\pi}_{3gn} \hat{o}_{3gn} \\
s' & : & \pi^r_{3gn} \pi_{3gn} \hat{\omega}_3 \\
s' & : & \hat{\pi}^r_{3gn} \hat{\pi}_{3gn} \hat{\omega}_3
\end{array}
$$

Intransitive verbs

$$
\begin{array}{lcl}
marche & : & \pi^r_{3\mathtt{s}} \, \boldsymbol{s} \\
marcher & : & \hat{\pi}^r \, \boldsymbol{i} \\
marché & : & \hat{\pi}^r \, \boldsymbol{p} \\
part & : & \pi^r_{3\mathtt{s}} \, \boldsymbol{s} \\
partir & : & \hat{\pi}^r \, \boldsymbol{i} \\
parti & : & \hat{\pi}^r_{\mathtt{ms}} \boldsymbol{p'} \\
partie & : & \hat{\pi}^r_{\mathtt{fs}} \boldsymbol{p'} \\
partis & : & \hat{\pi}^r_{\mathtt{mp}} \boldsymbol{p'} \\
parties & : & \hat{\pi}^r_{\mathtt{fp}} \boldsymbol{p'} \\
parles & : & \pi^r_{2\mathtt{s}} \, \boldsymbol{s} \, \omega^r \\
parles & : & \bar{\omega}^r \pi^r_{2\mathtt{s}} \, \boldsymbol{s} \\
parles & : & \hat{\omega}^r \pi^r_{2\mathtt{s}} \, \boldsymbol{s} \\
parlé & : & \hat{\pi}^r \, \boldsymbol{p} \, \omega^r \\
parlé & : & \hat{\omega}^r \hat{\pi}^r \, \boldsymbol{p}
\end{array}
$$

Transitive verbs			Auxiliaries and Modal verbs		
examine	:	$\pi^r_{3\mathtt{s}}\boldsymbol{s}o^\ell$			
examine	:	$\bar{o}^r\pi^r_{3\mathtt{s}}\boldsymbol{s}$			
examine	:	$\hat{o}^r\pi^r_{3\mathtt{s}}\boldsymbol{s}$	*a*	:	$o_{pgn}{}^r\pi^r_{3\mathtt{s}}\boldsymbol{sp}^\ell\hat{\pi}\hat{o}_{gn}$
examiner	:	$\hat{\pi}^r\boldsymbol{i}o^\ell$	*a*	:	$\pi^r_{3g\mathtt{s}}\boldsymbol{sp}^\ell\hat{\pi}_{3g\mathtt{s}}$
examiner	:	$\bar{o}^r\hat{\pi}^r\boldsymbol{i}$	*avoir*	:	$o_{pgn}{}^r\hat{\pi}^r\boldsymbol{ip}^\ell\hat{\pi}\hat{o}_{gn}$
examiner	:	$\hat{o}^r\hat{\pi}^r\boldsymbol{i}$	*avoir*	:	$\hat{\pi}^r\boldsymbol{ip}^\ell\hat{\pi}$
examiné	:	$\hat{\pi}^r\boldsymbol{p}o^\ell$	*est*	:	$\hat{o}^r_{3g\mathtt{s}}\pi^r_{3\mathtt{s}}\boldsymbol{sp}^\ell\hat{\pi}\hat{o}_{g\mathtt{s}}$
examiné	:	$\hat{o}^r_{\mathtt{ms}}\hat{\pi}^r\boldsymbol{p}$	*est*	:	$\pi^r_{3g\mathtt{s}}\boldsymbol{sp}'^\ell\hat{\pi}_{g\mathtt{s}}$
examiné	:	$\hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\tilde{\pi}^\ell$	*est*	:	$\pi^r_{3g\mathtt{s}}\boldsymbol{sp}^\ell\hat{o}_{g\mathtt{s}}$
examiné	:	$\hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi}$	*est*	:	$\hat{\omega}^r\pi^r_{3\mathtt{s}}\boldsymbol{sp}^\ell\hat{\pi}\hat{\omega}$
examinée	:	$\hat{o}^r_{\mathtt{fs}}\hat{\pi}^r\boldsymbol{p}$	*être*	:	$\hat{o}^r_{pgn}\hat{\pi}^r\boldsymbol{ip}^\ell\hat{\pi}\hat{o}_{gn}$
examinée	:	$\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell$	*être*	:	$\hat{\pi}^r_{pgn}\boldsymbol{ip}'^\ell\hat{\pi}_{pgn}$
examinée	:	$\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi}$	*être*	:	$\hat{\pi}^r_{pgn}\boldsymbol{ip}^\ell\hat{o}_{pgn}$
examinés	:	$\hat{o}^r_{\mathtt{mp}}\hat{\pi}^r\boldsymbol{p}$	*être*	:	$\hat{\omega}^r\hat{\pi}^r\boldsymbol{ip}^\ell\hat{\pi}\hat{\omega}$
examinés	:	$\hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\tilde{\pi}^\ell$	*doit*	:	$\pi^r_{3g\mathtt{s}}\boldsymbol{si}^\ell\hat{\pi}^r_{3g\mathtt{s}}$
examinés	:	$\hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi}$	*devoir*	:	$\hat{\pi}^r_{pgn}\boldsymbol{ii}^\ell\hat{\pi}_{pgn}$
examinées	:	$\hat{o}^r_{\mathtt{fp}}\hat{\pi}^r\boldsymbol{p}$	*dû*	:	$\hat{\pi}^r_{pgn}\boldsymbol{pi}^\ell\hat{\pi}_{pgn}$
examinées	:	$\hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\tilde{\pi}^\ell$			
examinées	:	$\hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\tilde{\pi}^\ell\tilde{\pi}$			

where p, g and n vary through all their possible values.
Basic types
$\boldsymbol{i}\,,\boldsymbol{p}\,,\boldsymbol{s}\,,$
$\nu_{gn}\to\pi_{3gn}\,,\ \pi_{pgn}\to\pi_{pn}\,,\ \hat{\pi}_{pgn}\to\hat{\pi}\,,\ \tilde{\pi}\,,$ for $p=1,2,3$, $g=\mathtt{m},\mathtt{f}$ and $n=\mathtt{s},\mathtt{p}$
$o_{pgn}\to\bar{o}\,,\ o\,,\ \hat{o}_{pgn}\to\hat{o}_{gn}\to\hat{o}\,,$ for $p=1,2,3$, $g=\mathtt{m},\mathtt{f}$ and $n=\mathtt{s},\mathtt{p}$
$\omega_{pn}\to\bar{\omega}_p\,,\hat{\omega}_p\to\hat{\omega}\,,$ for $p=1,2,3$
$\nu_{1g\mathtt{p}}\to o_{1g\mathtt{p}},\ \nu_{1g\mathtt{p}}\to\omega_{1\mathtt{p}}\,,\ \nu_{2g\mathtt{p}}\to o_{2g\mathtt{p}},\ \nu_{2g\mathtt{p}}\to\omega_{2\mathtt{p}}$ for $g=\mathtt{m},\mathtt{f}$

Bibliography

[1] Bargelli, Danièle and Joachim Lambek, An algebraic approach to French sentence structure, P. de Groote, G. Morrill, C. Retoré, eds., *Logical Aspects of Computational Linguistics 2001*, LNAI 2099, 62-78, 2001.

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

[3] Cardinaletti, Anna, Pronouns in Germanic and Romance Languages: An overview, in Hinrichs, Kathol and Nakazowa, eds., *Complex Predicates in Non-derivational Syntax, Syntax and Semantics*, Vol 30, Academic.

[4] Casadio, Claudia and Joachim Lambek, An algebraic analysis of clitic pronouns in Italian, P. de Groote, G. Morrill, C. Retoré, eds., *Logical Aspects of Computational Linguistics 2001*, LNAI 2099, 110-124, 2001.

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

[6] Lambek, Joachim, Type Grammar revisited, in: Alain Lecomte et al., eds., *Logical Aspects of Computational Linguistics*, Springer LNAI 1582, 1-27, 1999.

[7] Oehrle, Richard, A parsing algorithm for pregroup grammars, in: *Proceedings of Categorial Grammars 2004*, Montpellier France, 59-75, 2004.

[8] Pollard, Carl and Ivan A. Sag, Head-driven phrase structure grammar, The University of Chicago Press, 1994.

[9] Preller, Anne and Joachim Lambek, Free compact 2-categories, Mathematical Structures for Computer Sciences, 17(1), 1-32, Cambridge University Press, doi:10.1017/S0960129506005901, 2007.

[10] Preller, Anne, Toward Discourse Representation Via Pregroup Grammars, JoLLI, Vol. 16, 173-194, doi:10.1007/s10849-006-9033-y, 2007.

[11] Preller, Anne, Semantic pregroup grammars handle long distance dependencies in French, TALN 2007, *Workshop on 'Formalismes syntaxiques de haut niveau'*, Toulouse, France, June 5-8, 2007.

[12] Preller, Anne, Linear Processing with Pregroup Grammars, *Studia Logica*, vol. 87, issue 2:3, 171-197, 2007.