# Pregroup Grammars with Linear Parsing: Long Distance Dependency of Clitics in French

Anne Preller, Violaine Prince

# Pregroup grammars with linear parsing: long distance dependency of clitics in French

Anne Preller
Violaine Prince

**Abstract**

The linear parsing algorithm for pregroup grammars presented here exploits regularities of types in the dictionary. Sufficient conditions on the dictionary are given for the algorithm to be complete. Its working is illustrated by a grammar with distant agreement of features in French, including modal verbs, clitics, the compound past and the passive mode. The semantic interpretation of sentences parsed with this grammar is a predicate formula and can be computed from the parsing in time proportional to the size of the parsing.

*keywords*: Categorial grammars, pregroup grammars, linear parsing algorithm, distant dependencies, agreement of features, French clitics

## 1 Introduction

Pregroup grammars are introduced in [Lambek 1999] and simplify the earlier syntactic calculus of [Lambek 1958]. Pregroup grammars are lexical like other categorial grammars. A pregroup grammar consists of a dictionary and just 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 $X_l$ such that $v_l : X_l$ belongs to the dictionary for $1 \leq l \leq n$ and checks if successive applications of the generalized contraction rule reduce the concatenation $X_1 \dots X_n$ to the *sentence type* $\boldsymbol{s}$. For a fixed string $X_1 \dots X_n$, there may be several ways how to apply the generalized contraction rule. Each such choice of contractions is a *reduction of* $X_1 \dots X_n$ *to* $\boldsymbol{s}$ . A string of words $v_1 \dots v_n$ is recognized as a sentence, if there is at least one choice of types $X_1 \dots X_n$ and at least one reduction of $X_1 \dots X_n$ to $\boldsymbol{s}$. Each such reduction constitutes a parsing of $v_1 \dots v_n$.

Pregroup grammars have a cubic-time parsing algorithm which interweaves type assignment and type checking by processing the string of words from left to right. This algorithm does not take into account any properties specific to natural languages. In fact, it can be used as an algorithm for proof search in the theory of pregroups, [Degeilh-Preller]. Our believe is that humans process strings of words in linear time and that pregroups are versatile enough to simulate such processing. Strings of types from a dictionary for a natural language are not arbitrary. In this paper we formulate some of the properties such dictionaries may have and show that they are sufficient for a complete linear-time parsing algorithm.

The algorithm is illustrated by a grammar which handles long distance agreement of person, gender and number of the past participle with clitics in French. Clitics have been studied with pregroup grammars in French, [Bargelli-Lambek], and in Italian, [Casadio-Lambek], but without agreement. Our analysis differs from that given in [*loc. cit.*] for two reasons. First of all, we want to avoid the meta-rule used there and base the analysis inside an ordinary pregroup grammar. The other reason is that we prefer to think of clitics as designating individuals or sets of individuals, not operators on relations. Long distant dependency are captured by certain basic types called *shadows*. They are part of the semantical interpretation we extract from the words and as such are persistent throughout the sentence. As a side effect, they transmit long distant dependency. Whereas the other basic types represent grammatical notions like sentences, pronouns, etc, the shadows are implicit in the syntax and of anaphoric nature.

The next section briefly recalls some basic properties of pregroups and the rudiments of a semantical interpretation into predicate logic. The following section presents the sample grammar and in the last section, we define the parsing algorithm.

# 2 Basic notions

We briefly recall the definition of pregroups and the construction of a freely generated pregroup by [Lambek 1999]. Then we describe reductions geometrically as graphs and give the corresponding semantical interpretation into predicate logic. More about this interpretation following the lines of Discourse Representation Structures in [Kamp-Reyle] can be found in [Preller06].

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* $a^\ell$ and a *right adjoint* $a^r$ satisfying

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

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

One derives

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

The *free* pregroup $P(B)$ generated by a partially ordered set of *basic types* $B = \{..., a, b, ...\}$ is characterized in [Lambek 1999] as the free monoid generated from the set of *simple types* $\Sigma$ consisting of the *iterated adjoints* of basic types

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

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, we have

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

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

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

| | |
|---|---|
| (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 [Lambek 1999]. For example, consider the dictionary

| | | |
|---|---|---|
| *Marie* | : | $\pi_{3\mathtt{fs}}$ |
| *Marie* | : | $o$ |
| *Jean* | : | $\pi_{3\mathtt{ms}}$ |
| *Jean* | : | $o$ |
| *examine* | : | $\pi_{3\mathtt{s}}^r \boldsymbol{s} o^\ell$ |

The basic type $\pi_{3\mathtt{fs}}$ stands for 'subject third person feminine singular', or more generally, $\pi_{pgn}$ for 'subject of person $p$, gender $g$ and number $n$', where $p \in \{1, 2, 3\}$, $g \in \{\mathtt{m}, \mathtt{f}\}$ and $n \in \{\mathtt{s}, \mathtt{p}\}$. Here, `m` stands for 'masculine', `f` for 'feminine', `s` for 'singular' and `p` for 'plural'. We also have the basic types $\pi_{pn}$ for the subject when only the person and the number matter and $\pi$ when person, gender and number do not matter. The basic types $o$ and $\boldsymbol{s}$ stand for 'direct object' respectively for 'sentence in the present'. It is assumed that

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

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,

*Marie* *examine* *Jean*

(Mary examines John)

$(\pi_{3\mathtt{fs}})\ (\pi_{3\mathtt{s}}^r\, \boldsymbol{s}\, o^\ell)\ (o) \to \boldsymbol{s}$

This derivation is justified by the generalized contractions $\pi_{3\mathtt{fs}}\pi_{3\mathtt{s}}^r \to 1$ and $o^\ell o \to 1$. As customary, the types have been written under the words and the generalized contractions are indicated by under-links. In fact, the under-links uniquely determine the derivation. A systematic study of graphs as proofs in pregroups has been undertaken in [Preller-Lambek]. For our purposes here it suffices to remark that a derivation of $s_1 \dots s_n$ to a substring $s_{i_1} \dots s_{i_p}$ consisting of generalized contractions only is entirely determined by an *algebraic part* and a *geometrical part* $R$ called *reduction*. A *reduction* $R$ is a set of two-element subsets $\{i,k\} \subseteq \{1,\ldots,n\}$, called *under-links*, 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 there is no $l \in \{1,\ldots,p\}$ such that $i \leq i_l \leq k$ or $k \leq i_l \leq i$

if $\{i,k\}$, $\{j,m\} \in R$ and $i < j < k$, then $i < m < k$.

The algebraic part consists of the generalized contractions

$s_i s_k \to 1$, for $i < k$ such that $\{i,k\} \in R$

A reduction $R$ is called a *reduction from* $s_1 \ldots s_n$ *to* $s_{i_1} \ldots s_{i_p}$, written

$R : s_1 \ldots s_n \Rightarrow s_{i_1} \ldots s_{i_p}$

if all four conditions above hold. If the substring $s_{i_1} \ldots s_{i_p}$ cannot be contracted any further, it is called an *irreducible form* of $s_1 \ldots s_n$.

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 if a type has a unique irreducible form, there may be different reductions bringing it to that form, e.g. both

$a^\ell\ a\ a^\ell\ a\ a^r\ a$ and $a^\ell\ a\ a^\ell\ a\ a^r\ a$

are reductions to the empty string.

After associating relational or functional symbols to the entries in the dictionary, we construct a translation into predicate logic from a reduction to the sentence type. This is done by replacing each basic type of the chosen lexical entry by the expression given in the dictionary and substituting in each argument place the symbol linked to its corresponding right or left adjoint. In our example the transitive verb *examine* is interpreted as a binary relation. Looking at the type $\pi^r_{3\mathtt{s}} \boldsymbol{s} o^\ell$ of *examine*, the basic type $\boldsymbol{s}$ indicates that the lexical entry defines a relational symbol and the right and left adjoints of basic types determine the argument places. Here, ${\pi_{3\mathtt{s}}}^r$ correspond to the first argument place $x_1$ and $o^\ell$ to the second $x_2$. According to this convention, the types for proper names, which are just single basic types, do not introduce argument places and are translated by constants.

| | | | |
|---|---|---|---|
| *Marie* | : | $\pi_{3\mathtt{fs}}$ | `marie` |
| *Marie* | : | $o$ | `marie` |
| *Jean* | : | $\pi_{3\mathtt{ms}}$ | `jean` |
| *Jean* | : | $o$ | `jean` |
| *examine* | : | $\pi^r_{3\mathtt{s}} \boldsymbol{s} o^\ell$ | `examiner`$(x_1, x_2)$ |

Now the under-link from $\pi_{3\mathtt{fs}}$ to $\pi^r_{3\mathtt{s}}$ tells us that the constant `marie` corresponding to the basic type $\pi_{3\mathtt{fs}}$ occupies the first argument place $x_1$ corresponding to the right adjoint $\pi^r_{3\mathtt{s}}$. Similarly, the under-link from $o^\ell$ to $o$ puts the second constant `jean` into the second argument place. Hence the translation of

*Marie* *examine* *Jean*

$(\pi_{3\mathtt{fs}})\quad (\pi^r_{3\mathtt{s}}\ \boldsymbol{s}\ o^\ell)\quad (o)\quad \to \boldsymbol{s}$

becomes, after substitution,

`examiner(marie, jean)`.

More generally, according to [Preller06], the basic type(s) in a lexical entry are translated by functional or relational symbols. The argument places of these symbols are identified with right or left adjoints of basic types of the lexical entry. Each non basic type must be an argument place of at least one functional or relational symbol.

The translation of the sentence is constructed from the translations of the words by substitution. The translation then implies a characterizing formula of predicate logic, in the style of [Kamp-Reyle].

The insistence that every simple type of the entry must correspond to a symbol of the logic may force us to choose more involved types than needed for mere syntactic recognition. Suppose we added a new basic type $\boldsymbol{p}$ standing for the past participle and the lexical entries *examiné* : $\boldsymbol{p}\, o^{\ell}$ and *a* : $\pi_{3\mathtt{s}}^{r}\boldsymbol{s}\boldsymbol{p}^{\ell}$ to our dictionary. The augmented dictionary would recognize the sentence

*Jean* *a* *examiné* *Marie*
JOHN HAS EXAMINED MARY
$\pi_{3\mathtt{ms}}\ (\pi_{3\mathtt{s}}^{r}\,\boldsymbol{s}\,\boldsymbol{p}^{\ell})\ (\boldsymbol{p}\,o^{\ell})\quad (o)\quad .$

However, the entry *examiné* : $\boldsymbol{p}\, o^{\ell}$ would correspond to a unary relation. Surely, the relation translating a verb should depend on a stable number of variables in all its temporal aspects. And the semantic function of the auxiliary is to provide the temporal aspect rather than the missing argument place. If we add

| | | | |
|---|---|---|---|
| *examiné* | : | $\pi^{r}\boldsymbol{p}o^{\ell}$ | $\mathtt{examiner}(x_1, x_2)$ |
| *a* | : | $\pi_{3\mathtt{s}}^{r}\boldsymbol{s}\boldsymbol{p}^{\ell}\pi_{3\mathtt{s}}$ | $\mathtt{avoir}(y)\ \ \mathtt{id}(x)$ , |

we get the following parsing

*Jean* *a* *examiné* *Marie*
JOHN HAS EXAMINED MARY
$\pi_{3\mathtt{ms}}\ (\pi_{3\mathtt{s}}^{r}\ \boldsymbol{s}\ \boldsymbol{p}^{\ell}\,\pi_{3\mathtt{s}})\ (\pi^{r}\,\boldsymbol{p}\,o^{\ell})\quad (o)\, .$

Now we can correctly interpret the past participle by a binary relation, in fact the same we used for other forms of the same verb. The type for the auxiliary *a* has two basic types, namely $\boldsymbol{s}$ and $\pi_{3\mathtt{s}}$. Hence we translate the entry jointly by the predicate $\mathtt{avoir}(y)$ and the unary functional symbol $\mathtt{id}(x)$. The predicate symbol $\mathtt{avoir}$ translates the first basic type $\boldsymbol{s}$. Its argument-place $y$ corresponds to $\boldsymbol{p}^{\ell}$. The functional symbol $\mathtt{id}$ translates the second basic type $\pi_{3\mathtt{s}}$ with the argument-place $x$ given by $\pi_{3\mathtt{s}}^{r}$. A model will interpret $\mathtt{id}$ as the identity function, hence we impose the axiom

$\mathtt{id}(x) = x\, .$

The auxiliary verb *avoir* has the role of a temporal operator. Axioms could be added to the logic to express the temporal meaning, but this goes beyond the scope of our endeavor here. The interpretation of the sentence above is now obtained by substituting according to the under-links, i.e.

$\mathtt{avoir}(\mathtt{examiner}(\mathtt{id}(\mathtt{jean}), \mathtt{marie})).$

Using the equality $\mathtt{id}(\mathtt{jean}) = \mathtt{jean}$, we derive

$\mathtt{avoir}(\mathtt{examiner}(\mathtt{jean}, \mathtt{marie})).$

Similarly, the infinitive of a verb will have the same number of arguments than finite forms, i.e. the lexical entry with the semantic interpretation is

*examiner* : $\pi^{r}\boldsymbol{i}o^{\ell}$ $\quad\mathtt{examiner}(x_1, x_2)$

The motivation given above for the extra non-basic type in the infinitive or the past participle is semantical. We will see in the next section that it also serves for agreement of distant constituents in compound tenses or in the presence of modal verbs.

# 3 Agreement in the French sentence

In French, the personal pronoun in the role of a direct object precedes the verb. In the compound past of the active form, the past participle agrees in gender and number with the direct object clitic. If the verb is in passive form or forms its compound past with the auxiliary *être*, the past participle agrees in gender and number with the subject. Below, we will only consider clitics in the role of a direct object complement to the verb. An extension of the dictionary to include indirect object clitics is straightforward, but would extend this section beyond reasonable limits.

## 3.1 A dictionary with multiple entries

We add to the basic types of the preceding section new basic types for direct object clitics $o_{pgn}$, depending on the features of person $p = 1, 2, 3$, gender $g = \mathtt{m}, \mathtt{f}$ and number $n = \mathtt{s}, \mathtt{p}$. Moreover, they have 'shadows', $\hat{o}_{pgn}$ and $\hat{o}$, to capture distant dependencies. The types $\hat{o}_{gn}$ are used if the person does not matter, but gender and number do. The type of the clitic depends on the person because certain combinations of two clitics are impossible depending on the person, but we do not discuss this topic here. We assume

$\hat{o}_{pgn} \to \hat{o}_{gn} \to \hat{o}\,,\; o_{pgn} \to \hat{o}_{gn} \to \hat{o}\,,$

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

Next we will extend the dictionary to cover sentences like

| | |
|---|---|
| *Marie les examine* | *Marie s'examine* |
| (MARY EXAMINES THEM) | (MARY EXAMINES HERSELF) |
| *Marie les a examinés* | *Marie s'est examinée* |
| (MARY HAS EXAMINED THEM) | (MARY HAS EXAMINED HERSELF ) |
| *Marie doit les examiner* | *Marie doit s'examiner* |
| (MARY MUST EXAMINE THEM) | (MARY MUST EXAMINE HERSELF) |
| *Marie doit les avoir examinés* | *Marie doit s'être examinée* |
| (MARY MUST HAVE EXAMINED THEM) | (MARY MUST HAVE EXAMINED HERSELF) |
| *Marie est examinée par Jean* | *Marie est examinée* |
| (MARY IS EXAMINED BY JOHN) | (MARY IS EXAMINED) |

The lexical entries for personal pronoun *les* and the reflexive pronoun *s'* are

| | | | |
|---|---|---|---|
| *les* | : | $o_{3\mathtt{mp}}$ | $\mathtt{C}$ |
| *les* | : | $o_{3\mathtt{fp}}$ | $\mathtt{C}$ |
| $s'$ | : | $\pi_{3\mathtt{ms}}{}^{r}\pi_{3\mathtt{ms}}\hat{o}_{3\mathtt{ms}}$ | $\mathtt{id}(x)\ \mathtt{id}(x)$ |
| $s'$ | : | $\pi_{3\mathtt{fs}}{}^{r}\pi_{3\mathtt{fs}}\hat{o}_{3\mathtt{fs}}$ | $\mathtt{id}(x)\ \mathtt{id}(x)$ |
| $s'$ | : | $\pi_{3\mathtt{mp}}{}^{r}\pi_{3\mathtt{mp}}\hat{o}_{3\mathtt{mp}}$ | $\mathtt{id}(x)\ \mathtt{id}(x)$ |
| $s'$ | : | $\pi_{3\mathtt{fp}}{}^{r}\pi_{3\mathtt{fp}}\hat{o}_{3\mathtt{fp}}$ | $\mathtt{id}(x)\ \mathtt{id}(x)$ |

As the semantical translation does not depend on the features of gender and number, we may represent these six entries by the abbreviation

$$\begin{array}{lcll} les & : & o_{3g\mathtt{p}} & \mathtt{C} \\ s' & : & \pi^{r}_{3gn}\pi_{3gn}\hat{o}_{3gn} & \mathtt{id}(x)\ \ \mathtt{id}(x) \end{array}\quad \text{, where } g \in \{\mathtt{m}, \mathtt{f}\}\,,\; n \in \{\mathtt{s}, \mathtt{p}\}.$$

If the context permits, the set of values for the indices $p, g, n$ is omitted.

We remark that the lexical entries $s' : \pi^{r}_{3gn}\pi_{3gn}\hat{o}_{3gn}$ express a dependence on the gender and number of the subject and hand these features to the shadow object. This is achieved by using the same indices in $\pi^{r}_{3gn}$ and $\hat{o}_{3gn}$ . The anaphoric content is captured by the occurrence of two basic types, $\pi_{3gn}$ and $\hat{o}_{3gn}$. Both are translated by the unary functional symbol $\mathtt{id}$ and depend on the same argument-place $x$ corresponding to $\pi^{r}_{3gn}$. Recall that $id(x) = x$ holds in the logic.

In simple tenses, clitics do not require agreement with the following verb. Their preverbal position makes it necessary to assign a new type to the verb, added to the entries given in the former section and recalled here in parentheses.

$$\begin{array}{lcll} (\textit{examiner} & : & \pi^r \boldsymbol{i} \boldsymbol{o}^\ell & \mathtt{examiner}(x_1, x_2)) \\ \textit{examiner} & : & \hat{o}^r \pi^r \boldsymbol{i} & \mathtt{examiner}(y_1, y_2) \\ (\textit{examine} & : & \pi^r_{3\mathtt{s}} \boldsymbol{s} o^\ell & \mathtt{examiner}(x_1, x_2)) \\ \textit{examine} & : & \hat{o}^r \pi^r_{3\mathtt{s}} \boldsymbol{s} & \mathtt{examiner}(y_1, y_2) \end{array} .$$

In the new entries, the first variable $y_1$ corresponds to $\pi^r$ respectively $\pi^r_{3\mathtt{s}}$ and the second argument place $y_2$ to $\hat{o}^r$.

The gender of the pronoun *les* can be masculine or feminine in the sentence *Marie les examine* (MARY EXAMINES THEM). The two possible type assignments with a reduction to the sentence type reflect this fact. In opposition, only one of the four possible type assignments for the reflexive pronoun *s'* will do in the sentence *Marie s'examine* (MARY EXAMINES HERSELF). Hence, we find the following reductions

$$\begin{array}{lll} \textit{Marie} & \textit{les} & \textit{examine} \\ (\pi_{3\mathtt{fs}}) & (o_{3g\mathtt{p}}) & (\hat{o}^r \pi^r_{3\mathtt{s}}\ \boldsymbol{s}) \end{array} , \quad g = \mathtt{m}, \mathtt{f} \qquad \begin{array}{lll} \textit{Marie} & \textit{s'} & \textit{examine} \\ (\pi_{3\mathtt{fs}}) & (\pi^r_{3\mathtt{fs}}\ \pi_{3\mathtt{fs}}\, \hat{o}_{3\mathtt{fs}}) & (\hat{o}^r \pi^r_{3\mathtt{s}}\ \boldsymbol{s}) \end{array} .$$

As $g$ can take two values, the left hand display corresponds to two different type assignments, differing by $o_{3\mathtt{mp}}$ and $o_{3\mathtt{fp}}$ for the clitic *les*. The reduction itself remains unchanged, the set of links is the same for both type assignments. This observation is important when type assignments have to be chosen and tested for existence of reductions. The value of $g$ is irrelevant and we can find both reductions by computing just one. Note that the semantic difference between the left and right hand sentences above is correctly captured by the reductions

*Marie les examine* : $\mathtt{examiner}(\mathtt{marie}, \mathtt{C})$

*Marie s'examine* : $\mathtt{examiner}(\mathtt{id}(\mathtt{marie}), \mathtt{id}(\mathtt{marie}))$

As $\mathtt{id}(x) = x$, the latter translation is equivalent to

*Marie s'examine* : $\mathtt{examiner}(\mathtt{marie}, \mathtt{marie})$.

The type of the reflexive pronoun depends on the person to avoid non-sentences like **Tu s'examine*(YOU EXAMINE HIMSELF). Indeed, *tu* : $\pi_{2gs}$, $g = \mathtt{m}, \mathtt{f}$ and $\pi_{2g\mathtt{s}} \pi^r_{3g\mathtt{s}} \not\to 1$.

In the compound past, the clitic is separated from its verb by the auxiliary. The auxiliary does not show the relevant features by its form, but it carries them to the following word(s). The lexical entries below model this behavior by 'remembering' types.

$$\begin{array}{lcllll} \textit{avoir} & : & {o_{pgn}}^r \pi^r \boldsymbol{i}\boldsymbol{p}^\ell \pi \hat{o}_{gn} & \mathtt{avoir}(y) & \mathtt{id}(x_1) & \mathtt{id}(x_2) \\ \textit{a} & : & {o_{3gn}}^r \pi^r_{3\mathtt{s}} \boldsymbol{s}\boldsymbol{p}^\ell \pi \hat{o}_{gn} & \mathtt{avoir}(y) & \mathtt{id}(x_1) & \mathtt{id}(x_2) \\ \textit{être} & : & \hat{o}^r_{pgn} \pi^r \boldsymbol{i}\boldsymbol{p}^\ell \pi \hat{o}_{gn} & \mathtt{être}(y) & \mathtt{id}(x_1) & \mathtt{id}(x_2) \\ \textit{est} & : & \hat{o}^r_{3g\mathtt{s}} \pi^r_{3\mathtt{s}} \boldsymbol{s}\boldsymbol{p}^\ell \pi \hat{o}_{g\mathtt{s}} & \mathtt{être}(y) & \mathtt{id}(x_1) & \mathtt{id}(x_2) \\ \textit{examiné} & : & \hat{o}^r_{\mathtt{ms}} \pi^r \boldsymbol{p} & \mathtt{examiner}(x_1, x_2) & & \\ \textit{examinée} & : & \hat{o}^r_{\mathtt{fs}} \pi^r \boldsymbol{p} & \mathtt{examiner}(x_1, x_2) & & \\ \textit{examinés} & : & \hat{o}^r_{\mathtt{mp}} \pi^r \boldsymbol{p} & \mathtt{examiner}(x_1, x_2) & & \\ \textit{examinées} & : & \hat{o}^r_{\mathtt{fp}} \pi^r \boldsymbol{p} & \mathtt{examiner}(x_1, x_2) & & \end{array} .$$

The basic type $\pi$ in the first four entries above is translated by $\mathtt{id}(x_1)$, where in all entries the variable $x_1$ corresponds to the right adjoint $\pi^r$, with indices or without indices. Similarly, the basic types $\hat{o}_{gn}$ are translated by $\mathtt{id}(x_2)$, where $x_2$ corresponds to $o^r$, with the appropriate indices, with hat or without. The left adjoint $\boldsymbol{p}^\ell$ corresponds to the variable $y$.

Recall that the plural clitic *les* has two types, namely $o_{3\mathtt{mp}}$ and $o_{3\mathtt{fp}}$. If we choose the former, the past participle must be masculine plural. If we choose the latter, it must be feminine plural. However, both sentences have the same under-links, i.e. the same reduction. Though we do not know the correct choice of the value for $g$ until we see the last word, it does not matter when searching for a reduction.

*Marie les a examinés*

$$(\pi_{3\mathtt{fs}})\ (o_{3\mathtt{mp}})\ (o^r_{3\mathtt{mp}}\pi^r_{3\mathtt{s}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \pi\ \hat{o}_{\mathtt{mp}})\ (\hat{o}^r_{\mathtt{mp}}\pi^r\ \boldsymbol{p})$$

*Marie les a examinées*

$$(\pi_{3\mathtt{fs}})\ (o_{3\mathtt{fp}})\ (o^r_{3\mathtt{fp}}\pi^r_{3\mathtt{s}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \pi\ \hat{o}_{\mathtt{fp}})\ (\hat{o}^r_{\mathtt{fp}}\pi^r\ \boldsymbol{p})$$

If the clitic is a reflexive pronoun, the auxiliary in the compound tense is *être*. The past participle agrees in gender and number with the clitic if the latter is the direct object. Hence the type of *être* is similar to that of *avoir*, except that it is tailored to the reflexive pronoun, and therefore starts with $\hat{o}^r_{pgn}$ instead of $o^r_{pgn}$ .

*Marie s' est examinée*

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

Whereas the auxiliaries *avoir* and *être* 'remember' the features of the object, the modal verbs 'remember' the features of the subject. The clitic is positioned between the modal verb and the verb of which it is the object complement.

| | | | |
|---|---|---|---|
| *devoir* | : | $\pi^r_{pgn}\boldsymbol{i}\boldsymbol{i}^\ell\pi_{pgn}$ | $\mathtt{devoir}(y)\ \mathtt{id}(x)$ |
| *doit* | : | $\pi^r_{3gs}\boldsymbol{s}\boldsymbol{i}^\ell\pi^r_{3gs}$ | $\mathtt{devoir}(y)\ \mathtt{id}(x)$ |

where $p = 1, 2, 3;\ g = \mathtt{m}, \mathtt{f};\ n = \mathtt{s}, \mathtt{p}$. In these entries, the predicate symbol `devoir` translates the basic type $\boldsymbol{i}$ respectively $\boldsymbol{s}$. The unary functional symbol `id` translates the basic type $\pi_{pgn}$. The variable $y$ corresponds to $\boldsymbol{i}^\ell$ and $x$ to $\pi^r_{pgn}$. Then we have the two reductions to the sentence type

*Marie doit les examiner*

$$(\pi_{3\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \pi_{3\mathtt{fs}})\ (o_{3g\mathtt{p}})\ (\hat{o}^r\ \pi^r\ \boldsymbol{i})\text{, where } g = \mathtt{m}, \mathtt{f}.$$

*Marie doit s' examiner*

$$(\pi_{3\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \pi_{3\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \pi_{3\mathtt{fs}}\ \hat{o}_{3\mathtt{fs}})\ (\hat{o}^r\ \pi^r\ \boldsymbol{i})\ .$$

The translations of the two sentences are after replacement of `id(marie)` by `marie`

```
devoir(examiner(marie,C))
devoir(examiner(marie,marie)).
```

The reason why the type of the modal verbs depends on the gender becomes evident when they are used in combination with the compound past. For example

*Marie doit les avoir examinés*

$$(\pi_{3\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \pi_{3\mathtt{fs}})\ (o_{3\mathtt{mp}})\ (o^r_{3\mathtt{mp}}\pi^r\ \boldsymbol{i}\ \boldsymbol{p}^\ell\ \pi\ \hat{o}_{\mathtt{mp}})\ (\hat{o}^r_{\mathtt{mp}}\pi^r\ \boldsymbol{p})$$

and

*Marie doit s' être examinée*

$$(\pi_{3\mathtt{fs}})(\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \pi_{3\mathtt{fs}})(\pi^r_{3\mathtt{fs}}\ \pi_{3\mathtt{fs}}\ \hat{o}_{3\mathtt{fs}})(\hat{o}^r_{3\mathtt{fs}}\ \pi^r\ \boldsymbol{i}\ \boldsymbol{p}^\ell\ \pi\ \hat{o}_{\mathtt{fs}})(\hat{o}^r_{\mathtt{fs}}\ \pi^r\ \boldsymbol{p})\quad .$$

Note that the non-sentence **Marie doit s'être examiné* has no reduction to the sentence type as $\hat{o}_{\mathtt{fs}} \not\to \hat{o}_{\mathtt{ms}}$.

If we want to extend our language fragment to cover intransitive verbs forming the compound past with *être*, we add to the dictionary

| | | | | |
|---|---|---|---|---|
| *être* | : | $\pi^r_{gn}\boldsymbol{i}\boldsymbol{p}^\ell\pi_{gn}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ |
| *est* | : | $\pi^r_{3g\mathtt{s}}\boldsymbol{s}\boldsymbol{p}^\ell\pi_{g\mathtt{s}}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ |
| *partir* | : | $\pi^r\,\boldsymbol{i}$ | $\mathtt{partir}(x)$ | |
| *part* | : | $\pi^r_{3\mathtt{s}}\,\boldsymbol{s}$ | $\mathtt{partir}(x)$ | |
| *parti* | : | $\pi^r_{\mathtt{ms}}\boldsymbol{p}$ | $\mathtt{partir}(x)$ | , |
| *partie* | : | $\pi^r_{\mathtt{fs}}\boldsymbol{p}$ | $\mathtt{partir}(x)$ | |
| *partis* | : | $\pi^r_{\mathtt{mp}}\boldsymbol{p}$ | $\mathtt{partir}(x)$ | |
| *parties* | : | $\pi^r_{\mathtt{fp}}\boldsymbol{p}$ | $\mathtt{partir}(x)$ | |

where $g = \mathtt{m}, \mathtt{f};\ n = \mathtt{s}, \mathtt{p}$.

Then we find the following reduction to the sentence type

*Marie* *doit* *être* *partie*
MARY MUST HAVE LEFT

$$(\pi_{3\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{i}^\ell\ \pi_{3\mathtt{fs}})\ (\pi^r_{\mathtt{fs}}\ \boldsymbol{i}\ \boldsymbol{p}^\ell\ \pi_{\mathtt{fs}})\ (\pi^r_{\mathtt{fs}}\ \boldsymbol{p})\,.$$

For the passive form of transitive verbs yet more entries are needed. In particular, we introduce a new basic type $\hat{\pi}$ for the agent of the passive form, when introduced by the preposition *par*(BY):

| | | | | |
|---|---|---|---|---|
| *être* | : | $\pi^r_{gn}\boldsymbol{i}\boldsymbol{p}^\ell\hat{o}_{gn}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ |
| *est* | : | $\pi^r_{3g\mathtt{s}}\boldsymbol{s}\boldsymbol{p}^\ell\hat{o}_{g\mathtt{s}}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ |
| *examiné* | : | $\hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\hat{\pi}^\ell$ | $\mathtt{examiné}(x_1, x_2)$ | |
| *examinée* | : | $\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\hat{\pi}^\ell$ | $\mathtt{examiné}(x_1, x_2)$ | , where $g = \mathtt{m}, \mathtt{f}$, $n = \mathtt{s}, \mathtt{p}$. |
| *examinés* | : | $\hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\hat{\pi}^\ell$ | $\mathtt{examiné}(x_1, x_2)$ | |
| *examinées* | : | $\hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\hat{\pi}^\ell$ | $\mathtt{examiné}(x_1, x_2)$ | |
| *par* | : | $\hat{\pi}\pi^\ell$ | $\mathtt{id}(z)$ | |

In the translation $\mathtt{examiné}(x_1, x_2)$ of these entries for the past participle, $x_1$ corresponds to $\hat{\pi}^\ell$ and $x_2$ to $\hat{o}^r_{gn}$ . The relational symbol $\mathtt{examiné}$ is different from $\mathtt{examiner}$. The semantic connection between the passive and the infinitive of a transitive verb can be expressed by the non-logical axiom

$$\mathtt{être}(\mathtt{examiné}(x_1, x_2)) \Leftrightarrow \mathtt{examiner}(x_2, x_1)\,,$$

but this would be beyond the subject of this paper.

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

*Marie* *est* *examinée* *par* *Jean*

$$(\pi_{3\mathtt{fs}})(\pi^r_{3\mathtt{fs}}\ \boldsymbol{s}\ \boldsymbol{p}^\ell\ \hat{o}_{\mathtt{fs}})(\hat{o}^r_{\mathtt{fs}}\ \boldsymbol{p}\ \hat{\pi}^\ell)(\hat{\pi}\ o^\ell)(o)$$

This reduction defines the translation

$$\mathtt{être}(\mathtt{marie}, \mathtt{examiné}(\mathtt{jean}, \mathtt{marie}) \Leftrightarrow \mathtt{examiner}(\mathtt{jean}, \mathtt{marie})$$

Finally, if the agent of the passive is absent, like in *Marie est examinée*, the past participle will have yet another a type.

| | | | | |
|---|---|---|---|---|
| *examiné* | : | $\hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | $\mathtt{examiné}(x_1, x_2)$ | $\mathtt{C}$ |
| *examinée* | : | $\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | $\mathtt{examiné}(x_1, x_2)$ | $\mathtt{C}$ |
| *examinés* | : | $\hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | $\mathtt{examiné}(x_1, x_2)$ | $\mathtt{C}$ |
| *examinées* | : | $\hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | $\mathtt{examiné}(x_1, x_2)$ | $\mathtt{C}$ |

The constant $\mathtt{C}$ translates the basic type $\hat{\pi}$ of the lexical entries above and names the implicit agent of the passive form.

*Marie* *doit* *être* *examinée*

$$(\pi_{3\mathtt{fs}})\ (\pi^r_{3\mathtt{fs}}\boldsymbol{s}\,\boldsymbol{i}^\ell\ \pi_{3\mathtt{fs}})\ (\pi^r_{\mathtt{fs}}\boldsymbol{i}\ \boldsymbol{p}^\ell\ \hat{o}_{\mathtt{fs}})\ (\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\ \hat{\pi}^\ell\hat{\pi})\,.$$

Instead of giving the translation for the parsing structure above, we briefly describe a method how to find it without examining to many type assignments, in spite of the multiple entries for the same words. For easier use, the lexical entries are listed together below:

Basic types

$\pi_{pgn} \to \pi_{pn} \to \pi$ , $\hat{o}_{pgn} \to \hat{o}_{gn} \to \hat{o}$ , $o_{pgn} \to \hat{o}_{gn} \to \hat{o}$ , $o$ , $\boldsymbol{i}$ , $\boldsymbol{p}$ , $\boldsymbol{s}$ ,
for $p = 1, 2, 3$, $g = \mathtt{m}, \mathtt{f}$ and $n = \mathtt{s}, \mathtt{p}$.

Noun phrases, clitics, *par* (BY)

| | | | |
|---|---|---|---|
| *Marie* | : | $\pi_{3\mathtt{fs}}$ | `marie` |
| *Marie* | : | $o$ | `marie` |
| *Jean* | : | $\pi_{3\mathtt{ms}}$ | `jean` |
| *Jean* | : | $o$ | `jean` |
| *les* | : | $o_{3gp}$ | `C` |
| *s'* | : | $\pi^r_{3gn}\pi_{3gn}\hat{o}_{3gn}$ | `id`$(x)$ `id`$(x)$ |
| *par* | : | $\hat{\pi}o^\ell$ | `id`$(z)$ |

Intransitive verbs

| | | | |
|---|---|---|---|
| *partir* | : | $\pi^r\,\boldsymbol{i}$ | `partir`$(x)$ |
| *parti* | : | $\pi^r_{\mathtt{ms}}\boldsymbol{p}$ | `partir`$(x)$ |
| *partie* | : | $\pi^r_{\mathtt{fs}}\boldsymbol{p}$ | `partir`$(x)$ |
| *partis* | : | $\pi^r_{\mathtt{mp}}\boldsymbol{p}$ | `partir`$(x)$ |
| *parties* | : | $\pi^r_{\mathtt{fp}}\boldsymbol{p}$ | `partir`$(x)$ |

Transitive verbs

| | | | |
|---|---|---|---|
| *examiner* | : | $\pi^r\boldsymbol{i}o^\ell$ | `examiner`$(x_1, x_2)$ |
| *examiner* | : | $\hat{o}^r\pi^r\boldsymbol{i}$ | `examiner`$(y_1, y_2)$ |
| *examine* | : | $\pi^r_{3\mathtt{s}}\boldsymbol{s}o^\ell$ | `examiner`$(x_1, x_2)$ |
| *examine* | : | $\hat{o}^r\pi^r_{3\mathtt{s}}\boldsymbol{s}$ | `examiner`$(y_1, y_2)$ |
| *examiné* | : | $\pi^r\boldsymbol{p}o^\ell$ | `examiner`$(x_1, x_2)$ |
| *examiné* | : | $\hat{o}^r_{\mathtt{ms}}\pi^r\boldsymbol{p}$ | `examiner`$(x_1, x_2)$ |
| *examiné* | : | $\hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\hat{\pi}^\ell$ | `examiné`$(x_1, x_2)$ |
| *examiné* | : | $\hat{o}^r_{\mathtt{ms}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | `examiné`$(x_1, x_2)$ `C` |
| *examinée* | : | $\hat{o}^r_{\mathtt{fs}}\pi^r\boldsymbol{p}$ | `examiner`$(x_1, x_2)$ |
| *examinée* | : | $\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\hat{\pi}^\ell$ | `examiné`$(x_1, x_2)$ |
| *examinée* | : | $\hat{o}^r_{\mathtt{fs}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | `examiné`$(x_1, x_2)$ `C` |
| *examinés* | : | $\hat{o}^r_{\mathtt{mp}}\pi^r\boldsymbol{p}$ | `examiner`$(x_1, x_2)$ |
| *examinés* | : | $\hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\hat{\pi}^\ell$ | `examiné`$(x_1, x_2)$ |
| *examinés* | : | $\hat{o}^r_{\mathtt{mp}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | `examiné`$(x_1, x_2)$ `C` |
| *examinées* | : | $\hat{o}^r_{\mathtt{fp}}\pi^r\boldsymbol{p}$ | `examiner`$(x_1, x_2)$ |
| *examinées* | : | $\hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\hat{\pi}^\ell$ | `examiné`$(x_1, x_2)$ |
| *examinées* | : | $\hat{o}^r_{\mathtt{fp}}\boldsymbol{p}\hat{\pi}^\ell\hat{\pi}$ | `examiné`$(x_1, x_2)$ `C` |

Auxiliaries

| | | | | | |
|---|---|---|---|---|---|
| *avoir* | : | $\pi^r \boldsymbol{ip}^\ell \pi$ | $\mathtt{avoir}(y)$ | $\mathtt{id}(x)$ | |
| *avoir* | : | ${o_{pgn}}^r \pi^r \boldsymbol{ip}^\ell \pi \hat{o}_{gn}$ | $\mathtt{avoir}(y)$ | $\mathtt{id}(x_1)$ | $\mathtt{id}(x_2)$ |
| *a* | : | $\pi^r_{3\mathtt{s}} \boldsymbol{sp}^\ell \pi$ | $\mathtt{avoir}(y)$ | $\mathtt{id}(x)$ | |
| *a* | : | ${o_{3gn}}^r \pi^r_{3\mathtt{s}} \boldsymbol{sp}^\ell \pi \hat{o}_{gn}$ | $\mathtt{avoir}(y)$ | $\mathtt{id}(x_1)$ | $\mathtt{id}(x_2)$ |
| *être* | : | $\pi^r_{gn} \boldsymbol{ip}^\ell \pi_{gn}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ | |
| *être* | : | $\pi^r_{gn} \boldsymbol{ip}^\ell \hat{o}_{gn}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ | |
| *être* | : | $\hat{o}^r_{pgn} \pi^r \boldsymbol{ip}^\ell \pi \hat{o}_{gn}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x_1)$ | $\mathtt{id}(x_2)$ |
| *est* | : | $\pi^r_{3g\mathtt{s}} \boldsymbol{sp}^\ell \pi_{g\mathtt{s}}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ | |
| *est* | : | $\pi^r_{3g\mathtt{s}} \boldsymbol{sp}^\ell \hat{o}_{g\mathtt{s}}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x)$ | |
| *est* | : | $\hat{o}^r_{3g\mathtt{s}} \pi^r_{3\mathtt{s}} \boldsymbol{sp}^\ell \pi \hat{o}_{g\mathtt{s}}$ | $\mathtt{être}(y)$ | $\mathtt{id}(x_1)$ | $\mathtt{id}(x_2)$ |

Modal verbs

| | | | | |
|---|---|---|---|---|
| *devoir* | : | $\pi^r_{pgn} \boldsymbol{ii}^\ell \pi_{pgn}$ | $\mathtt{devoir}(y)$ | $\mathtt{id}(x)$ |
| *doit* | : | $\pi^r_{3g\mathtt{s}} \boldsymbol{si}^\ell \pi^r_{3g\mathtt{s}}$ | $\mathtt{devoir}(y)$ | $\mathtt{id}(x)$ |

where $p = 1, 2, 3$, $g = \mathtt{m}, \mathtt{f}$ and $n = \mathtt{s}, \mathtt{p}$.

## 3.2 Choosing type assignments

The various types associated to a word by the dictionary present certain regularities. These can be exploited to avoid type assignments that cannot lead to a reduction to the sentence type. We illustrate this on the example sentence *Marie doit être examinée.* The sentence is processed as we hear it, word after word.

The first word *Marie* has two entries in the dictionary. After hearing *Marie*, we know that every type assignment will start with

$\pi_{3\mathtt{fs}}$ and $o$.

However, the dictionary has no occurrence of the simple type $o^r$, hence no matter how we continue, a string of simple types starting with $o$ has no reduction to the sentence type. Hence we can ignore this loosing type assignment when processing the following of words. The next word has two lexical entries, namely *doit* : $\pi^r_{3g\mathtt{s}} \boldsymbol{si}^\ell \pi_{3g\mathtt{s}}$, $g = \mathtt{m}, \mathtt{f}$. The type assignments for *Marie doit* to be processed are

$\pi_{3\mathtt{fs}}\ \pi^r_{3\mathtt{ms}} \boldsymbol{si}^\ell \pi_{3\mathtt{ms}}$ and $\pi_{3\mathtt{fs}}\ \pi^r_{3\mathtt{fs}} \boldsymbol{si}^\ell \pi_{3\mathtt{ms}}$.

The first is a loser, because it contains an irreducible substring ending in a right adjoint, namely $\pi_{3\mathtt{fs}}\ \pi^r_{3\mathtt{ms}}$. Indeed, there are no double right adjoints in the dictionary, the simple type $\pi^r_{3\mathtt{ms}}$ can only be contracted with a basic type to its left. Hence, which ever words we hear after *doit*, none of them will have a type assignment which eliminates $\pi^r_{3\mathtt{ms}}$. Therefore, after hearing *Marie doit*, we know that the only type assignment which might produce a reduction to the sentence type is $\pi_{3\mathtt{fs}}\, \pi^r_{3\mathtt{fs}} \boldsymbol{si}^\ell \pi_{3\mathtt{fs}}$. As we do not want to overburden the memory, we make the contraction $\pi_{3\mathtt{fs}} \pi^r_{3\mathtt{fs}} \to 1$ and store the result

$\boldsymbol{si}^\ell \pi_{3\mathtt{fs}}$

There are twenty possible entries for the next word, namely

$$\begin{array}{lll} \textit{être} & : & \hat{o}^r_{pgn} \pi^r \boldsymbol{ip}^\ell \pi \hat{o}_{gn}, \\ \textit{être} & : & \pi^r_{gn} \boldsymbol{ip}^\ell \pi_{gn}, \qquad p = 1, 2, 3;\ g = \mathtt{m}, \mathtt{f};\ n = \mathtt{s}, \mathtt{p}. \\ \textit{être} & : & \pi^r_{gn} \boldsymbol{ip}^\ell \hat{o}_{gn}, \end{array}$$

The last simple type of the stored string is $\pi_{3\mathtt{fs}}$. Comparing it with the first simple type of the entries for *être*, we find that $\pi_{3\mathtt{fs}} \hat{o}^r_{pgn}$ forms an irreducible string for all values of $p, g$ and $n$. Hence we can eliminate the twelve entries starting with $\hat{o}^r_{pgn}$. Again, out of the eight remaining entries, only two will not create an irreducible substring, namely those for which $g = \mathtt{f}$ and $n = \mathtt{s}$. Hence we compute the two possibly non-loosing assignments for *Marie doit être*

$\boldsymbol{s}\boldsymbol{i}^{\ell}\pi_{3\mathtt{fs}}\pi_{\mathtt{fs}}^{r}\boldsymbol{i}\boldsymbol{p}^{\ell}\pi_{\mathtt{fs}} \rightarrow \boldsymbol{s}\boldsymbol{p}^{\ell}\pi_{\mathtt{fs}}$ and $\boldsymbol{s}\boldsymbol{i}^{\ell}\pi_{3\mathtt{fs}}\pi_{\mathtt{fs}}^{r}\boldsymbol{i}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}} \rightarrow \boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}$ and store the results, namely

$\boldsymbol{s}\boldsymbol{p}^{\ell}\pi_{\mathtt{fs}}$ and $\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}$.

The last word *examinée* has three entries in the dictionary, namely

*examinée* : $\hat{o}_{\mathtt{fs}}^{r}\pi^{r}\boldsymbol{p}$
*examinée* : $\hat{o}_{\mathtt{fs}}^{r}\boldsymbol{p}\hat{\pi}^{\ell}$ .
*examinée* : $\hat{o}_{\mathtt{fs}}^{r}\boldsymbol{p}\hat{\pi}^{\ell}\hat{\pi}$

The first of the stored strings ends with $\pi_{\mathtt{fs}}$, hence would create the irreducible string $\pi_{\mathtt{fs}}\hat{o}_{\mathtt{fs}}^{r}$ with all three entries. So we may erase it from the memory. Remains the string

$\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}$

in the memory. Choosing the first entry for *examinée*, we contract

$\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}\hat{o}_{\mathtt{fs}}^{r}\pi^{r}\boldsymbol{p} \rightarrow \boldsymbol{s}\boldsymbol{p}^{\ell}\pi^{r}\boldsymbol{p}$ ,

we find the irreducible substring $p^{\ell}\pi^{r}$ and therefore eliminate the first entry as a possible type assignment. Next, we compare the stored string with the the second entry $\hat{o}_{\mathtt{fs}}^{r}\boldsymbol{p}\hat{\pi}^{\ell}$ and find

$\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}\hat{o}_{\mathtt{fs}}^{r}\boldsymbol{p}\hat{\pi}^{\ell} \rightarrow \boldsymbol{s}\pi^{\ell}$ .

As we are at the end of the processed string of word nothing else can be done.

Finally, concatenating the stored type $\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}$ with the last choice for *examinée*, namely $\hat{o}_{\mathtt{fs}}^{r}\boldsymbol{p}\hat{\pi}^{\ell}\hat{\pi}$, we find that it has a reduction to the sentence type. In fact, we can find this result reading the type from the dictionary from left to right, simple type by simple type, adding each simple type read to the memorized type. If the read type contracts with the last simple type in the memory we eliminate both and proceed. Else, we just proceed.

$\boldsymbol{s}\boldsymbol{p}^{\ell}\hat{o}_{\mathtt{fs}}\hat{o}_{\mathtt{fs}}^{r} \rightarrow \boldsymbol{s}\boldsymbol{p}^{\ell}$
$\boldsymbol{s}\boldsymbol{p}^{\ell}\boldsymbol{p} \rightarrow \boldsymbol{s}$
$\boldsymbol{s}\hat{\pi}^{\ell}$
$\boldsymbol{s}\hat{\pi}^{\ell}\hat{\pi} \rightarrow \boldsymbol{s}$ .

We describe this linear algorithm in the next section more formally and give sufficient conditions on the dictionaries for which it is complete.

# 4 Linear assignment

Certain properties of the sample grammar in the preceding section are sufficient conditions for a linear parsing algorithm. They are of two different kinds. One is an assumption about the form of the types in the dictionary. The other one assumes that the temporary memory does not get overcharged during processing. The idea that the temporary memory is limited to about seven bits is based on a study of [Miller] and illustrated convincingly with a linguistic example by [Lambek 2006]. We limit processing by assuming that dictionaries are *concise* (see below).

In the following, the partially ordered set $B$ and the free pregroup $P(B)$ generated by $B$ are fixed. As usual, a *dictionary $D$ over $B$* for a set of *words* $V$ is a map from $V$ to the set of subsets of $P(B)$. Instead of $T_l \in D(v_l)$ we may write $v_l : T_l$. We distinguish a basic type $\boldsymbol{s}$ called *sentence type.* A string of types $T_1 \ldots T_n$ is called a *type assignment* for $v_1 \ldots v_n$ if $v_l : T_l$ is an entry of the dictionary for $1 \leq l \leq n$. Searching for a reduction of $T_1 \ldots T_n$ to $\boldsymbol{s}$ is *called type checking.* A *parsing* of a string $v_1 ... v_n$ consists of a type assignment $T_l \in D(v_l)$ and a reduction of $T_1 ... T_n$ to $\boldsymbol{s}$.

We begin by describing an algorithm which combines search for reduction with type assignment. This algorithm is extracted from [Preller07] and reproduced here for completeness sake. It processes by stages reading the string of words $v_1 \dots v_n$ from left to right. At each stage, it either chooses a type for the word under examination or processes the assigned type by reading its simple types from left to right. The result is a reduction of the type processed so far to an irreducible type.

The set of *stages* associated to $v_1 \dots v_n$ consists of triples $s = (l, T_1 \dots T_l, p)$ where
$l$ is the number of the word $v_l$ being processed
$T_k = f_{k1} \dots f_{kq_k}$ in $D(v_k),\ 1 \le k \le l$, a type assignment for $v_1 \dots v_l$
$p$ a *position*, $0 \le p \le q_l$.

The stages are partially ordered as follows

$$(l, T_1 \dots T_l, p) \le (l', T'_1 \dots T'_{l'}, p') \Leftrightarrow l \le l',\ p \le p',\ T_k = T'_k \text{ for } 1 \le k \le l\,.$$

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

We remark that all stages $s$ except the initial have a unique immediate predecessor, which we denote by $s - 1$, i.e.

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

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 define the actual position $p(s)$ and the type read at this position $f_{p(s)}$. At the initial stage $p(s_{in}) = 0,\ f_{p(s_{in})} = 1$. A stage of the form $(l, T_1 \dots T_l, 0)$, $1 \le l \le n$, is called a *downloading stage* and serves to choose a type $T_l \in D(v_l)$ as soon as the word $v_l$ has been given. 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 $s = (l, T_1 \dots T_l, p)$, $p \ge 1$, 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

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

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

$$f_r = f_{kp'}\,.$$

The simple type $f_{p(s)}$ 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 $f_{p(s)}$ 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 $r = q_1 + \cdots + q_k + p'$.

**Definition 1.** ***Parsing Algorithm***
▼ *At the initial stage, let*

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

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

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

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

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

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

**Lemma 1.** *For every stage $s = (l, T_1 \ldots T_l, p)$, let the types $T(s)$ and $\overline{S(s)}$ be defined as follows:*

$$\begin{array}{lcl} T(s_{in}) & = & \overline{S(s_{in})} = 1 = f_0 \\ T(s) & = & T(s-1) f_{p(s)} \\ \overline{S(s)} & = & \begin{cases} \overline{S(s-1)}, & \text{if } s \text{ is downloading} \\ \overline{pop(S(s-1))}, & \text{else if } f_{t(s-1)} f_{p(s)} \to 1 \\ \overline{S(s-1)} f_{p(s)}, & \text{else} \end{cases} \end{array} .$$

*Then $T(s) = T_1 \ldots T_{l-1} f_{l1} \ldots f_{lp}$, for $1 \leq l \leq n$ and $1 \leq p \leq q_l$. Moreover, 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 $s = (l, T_1 \ldots T_l, p)$ is a particular case of the type checking algorithm in [Preller07], applied to the type assignment $T_1 \ldots T_l$. The property follows now from Theorem 6.5 in [*loc. cit.*].

Let $T_1 \ldots T_n$ be a type assignment of $v_1 \ldots v_n$ and consider a final stage $s = (n, T_1 \ldots T_n, q_n)$. According to the lemma above, $R(s)$ is a reduction to an irreducible form of $T_1 \ldots T_n$. 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 $T_1 \ldots T_n$ 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 2. Linearity**
A critical triple *of a string of simple types $t_1 \ldots t_q$ is a substring $t_i \ldots t_j \ldots t_k$ such that $i < j < k$ and*

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

*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.*

For example, the dictionary of the preceding section is linear. Indeed, the basic types of a fixed connected component appear either with with right adjoints or with left adjoints in the dictionary, but never with both a right and left adjoint. Hence no string of words has a type assignment containing a critical triple.

**Theorem 1** (**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.*

*Proof*: By Lemmas 5.3 and 5.4 in [Preller07], every linear string has a unique irreducible form and a unique reduction to this irreducible form.

The algorithm is complete under the assumption that it computes all final stages. However, many 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 3.** *(***Loosing stages)**
*A simple type $f$ is* right cancellable *in $D$ , if it is the sentence type $\boldsymbol{s}$ or if there is a simple type $f'$ such that $f \to f'$ and $f'^r$ occurs in $D$ . A stage $s = (l+1, T_1 \dots T_{l+1}, q_{l+1})$ associated to a string of words $v_1 \dots v_{l+1}$ , $1 \leq l < n$ , is* loosing *if for some position $i$ stored in $S(s)$, the simple type $f_i$ is not right cancellable.*

**Corollary 2.** *Assume that the dictionary is linear and let $t(s)$ denote the top of the stack $S(s)$. If at stage $s$ the simple type $f_{t(s)}$ is not right cancellable, then no final stage extending $s$ produces a reduction to the sentence type $\boldsymbol{s}$ .*

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

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. As there are as many final stages as there are different type assignments it is quite unlikely that a human will consider them all. The choice of the type for the next word depends on the meanings of the previous words, i.e. the already chosen types. We may assume that the selection criterion is good enough to keep the number of possible meanings manageable. For pregroup dictionaries, we can formulate such a criterion with the help of the Parsing Algorithm.

**Definition 4.** *(***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 loosing.*

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. The following is a simplification of Lemma 6.6. of [Preller07] to the linear case:

**Lemma 2** (**Constant number of basic steps per word**). *Let $D$ be a concise dictionary and $k_0$ bound the number of types per word and the length of all 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-loosing stages $(l, T_1 \dots T_l, q_l)$ is bounded by $k_0$. Moreover, the number of basic steps performed 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 computation steps for every simple type of $T_1$ is bounded by a constant, say $c$. When processing word $v_{l+1}$, only the non-loosing stages $(l, T_1 \dots T_l, q_l)$ are extended by a lexical entry $T_{l+1} \in D(v_l)$. By induction hypothesis, there are at most $k_0$ of the former and, by assumption, there are at most $k_0$ of the latter. 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 most $ck_0$ basic steps. The test performed at stage $s = (l+1, T_1 \dots T_l T_{l+1}, p)$ permits to recognize it as loosing if $f_{p(s)}$ is not right cancellable and $f_{t(s)} f_{p(s)} \not\to 1$. For a given $T_{l+1} \in D(v_l)$, at most one of the non-loosing stages $(l, T_1 \dots T_l, q_l)$ will yield a non-loosing stage $(l+1, T_1 \dots T_l T_{l+1}, q_{l+1})$. Hence after performing at most $ck_0^3$ basic steps for word $v_{l+1}$, at most $k_0$ stages $(l+1, T_1 \dots T_l T_{l+1}, q_{l+1})$ will be recognized as non-loosing.

**Theorem 3** (**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, finds the reductions to the sentence type.*

*Proof*: By the Completeness Theorem, the Parsing Algorithm finds all reductions to the sentence type of a given string of words. Every regroup grammar has a finite dictionary and therefore a bound for the number of types per word and the length of all types in the dictionary. By the Failure Detection Lemma, the number of basic steps performed for each word is bound by a constant. A control can be prevent the Parsing Algorithm to continue computation for loosing stages. The modified algorithm is still complete.

# 5 Conclusion

Our aim here was to illustrate that concepts from other grammars, for example HPSG's of [Pollard-Sag], can and should be used in pregroup grammars, but only to increase efficiency. The fact to use features in pregroup grammars does not imply that the grammar has a structure beyond the original concept. All we have done is to single out properties of the types in the dictionary which make parsing linear. If the properties were to be ignored, the grammar would not change, it would generate the same sentences with the same parsings and same semantical interpretation. The examples show that linear parsing may come at the cost of adding basic types or increasing the number of types per word in the dictionary. The contribution of this paper is to show that an explosion of basic types does not increase runtime but may rather 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 runtime is proportional. This constant can be lowered by exploiting certain regularities of features which make it possible to construct several distinct reductions with one and the same computation. Future work will go in this direction. Larger and more expressive language fragments are handled by non-linear dictionaries that also can be parsed in linear time as shown in [Preller07]. Our belief is that clitics in general, see [Cardinaletti], can be handled by pregroup grammars in a very efficient way.

## References


[Bargelli-Lambek] Daniéle Bargelli, Joachim Lambek, An algebraic approach to French sentence structure, P. de Groote, G. MOrrill, C. Retoré,*eds.*: LACL 2001, LNAI 2099, pp. 62-78, 2001

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

[Cardinaletti] Anna Cardinaletti, 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

[Casadio-Lambek] Claudia Casadio, Joachim Lambek, An algebraic analysis of clitic pronouns in Italian, P. de Groote, G. MOrrill, C. Retoré,*eds.*: LACL 2001, LNAI 2099, pp. 110-124, 2001

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

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

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

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

[Lambek 2006] Joachim Lambek, Reflections on English personal pronouns, preprint, McGill University, Montreal, 2006

[Miller] G.A. Miller, The magical number seven plus or minus two: Some limits to our capacity for processing information, Psychological Review 63, pp. 81-97, 1956

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

[Preller-Lambek] Anne Preller, Joachim Lambek, Free compact 2-categories, to appear in: Mathematical Structures for Computer Sciences, Cambridge University Press, pp. XX-YY, 2007

[Preller06] Anne Preller, Toward Discourse Representation Via Pregroup Grammars, to appear in JoLLI, reprint at: Laboratoire d'Informatique, Robotique et Microlectronique de Montpellier, Rapport de recherche 12948, 2006

[Preller07] Anne Preller, Linear processing with pregroups, Studia Logica, pp. XX-YY, 2007

Anne Preller Violaine Prince
preller@lirmm.fr prince@lirmm.fr
LIRMM, 161, RUE ADA, MONTPELLIER, FRANCE