Interaction Grammars

Bruno Guillaume, Guy Perrier

Interaction Grammars

Bruno Guillaume
LORIA, INRIA
Bruno.Guillaume@loria.fr

Guy Perrier
LORIA, Université Nancy2
Guy.Perrier@loria.fr

June 16, 2008

Abstract

Interaction Grammar (IG) is a grammatical formalism based on the notion of polarity. Polarities express the resource sensitivity of natural languages by modelling the distinction between saturated and unsaturated syntactic structures. Syntactic composition is represented as a chemical reaction guided by the saturation of polarities. It is expressed in a model-theoretic framework where grammars are constraint systems using the notion of tree description and parsing appears as a process of building tree description models satisfying criteria of saturation and minimality.

Keywords Grammatical formalism, Categorial Grammar, Unification, Polarity, Tree description

Introduction

Interaction Grammar (IG) is a grammatical formalism based on an old idea of O. Jespersen [20], L. Tesnière [46] and K. Adjukiewicz [2]: a sentence is viewed as a molecule with its words as the atoms; every word is equipped with a valence which expresses its capacity of interaction with other words, so that syntactic composition appears as a chemical reaction.

The first grammatical formalism that exploited this idea was Categorial Grammar (CG) [39]. In CG, constituents are equipped with types, which express their interaction ability in terms of syntactic categories. A way of highlighting this originality is to use polarities: syntactic types can be represented by partially specified syntactic trees, which are decorated with polarities that express a property of non saturation; a positive node represents an available grammatical constituent whereas a negative node represents an expected grammatical constituent; negative nodes tend to merge with positive nodes of the same type and this mechanism of neutralization between opposite polarities drives the composition of syntactic trees to produce saturated trees in which all polarities have been neutralized.

The notion of polarity in this sense was not used explicitly in computational linguistics until recently. To our knowledge, A. Nasr was the first to propose a formalism using polarized structures [31]. Then, nearly at the same time, R. Muskens [30], D. Duchier and S. Thater [15], and G. Perrier [33] proposed grammatical formalisms using polarities. The latter was a first version of IG, presented in the framework of linear logic. This version, which covers only the syntax of natural languages, was extended to the semantics of natural languages [35]. Then, S. Kahane showed that all well known formalisms (CFG, TAG, HPSG, LFG) can be viewed as polarized formalisms [21]. Unlike the previous approaches, polarities are used in a non monotonous way in Minimalist Grammar (MG). E. Stabler [43] proposes a formalization of MG which highlights this. Polarities are associated with syntactic features to control movement inside syntactic structures: strong features are used to drive the movement of phonetic forms (overt movement) and weak features are used to drive the movement of logical forms (covert movement).

With IG, we highlighted the fundamental mechanism of saturation between polarities underlying CG in a more refined way, because polarities are attached to the features used to describe constituents and not to the constituents themselves — but the essential difference lies in the change of framework: CG are

usually formalized in a generative deductive framework, the heart of which is the Lambek Calculus [23], whereas IG is formalized in a model-theoretic framework. A particular interaction grammar appears as a set of constraints, and parsing a sentence with such a grammar reduces to solving a constraint satisfaction problem. G. K. Pullum and B. C. Scholz highlighted the advantages of this change of framework [37]. Here, we are especially interested in some of these advantages:

- syntactic objects are tree descriptions which combine independent elementary properties in a very flexible way to represent families of syntactic trees;
- underspecification can be represented in a natural way by tree descriptions;
- partially well-formed sentences have a syntactic representation in the sense that, even if they have no complete parse trees, they can be characterized by tree descriptions.

The notion of tree description, which is central in this approach, was introduced by M. Marcus, D. Hindle and M. Fleck to reduce non-determinism in the parsing of natural languages [27]. It was used again by K. Vijay-Shanker to represent the adjoining operation of TAG in a monotonous way [48]. Then, it was studied systematically from a mathematical point of view [40] and it gave rise to new grammatical formalisms [22, 38].

If model theory provides a declarative framework for IG, polarities provide a step by step operational method to build models of tree descriptions: partially specified trees are superposed[1] under the control of polarities; some nodes are merged in order to saturate their polarities and the process ends when all polarities are saturated. At that time, the resulting description represents a completely specified syntactic tree. The ability of the formalism to superpose trees is very important for its expressiveness. Moreover, the control of superposition by polarities is interesting for computational efficiency.

In natural languages, syntax is a way to access semantics and a linguistic formalism worthy of the name must take this idea into account. If the goal of the article is to give a formal presentation of IG which focuses on the syntactic level of natural languages, the formalism is designed in such a way that various formalizations of semantics can be plugged into IG. The reader can find a first proposal in [35].

An important concern with IG is to provide a realistic formalism, which can be experimented parsing actual corpora. In order to combine the theoretical development of the formalism with experimentation, we have designed a parser, LEOPAR, based on IG [5]. If a relatively efficient parser is a first condition to get a realistic formalism, a second condition is to be able to build large coverage grammars and lexicons. With an appropriate tool, XMG [14], we have built a French interaction grammar with a relatively large coverage [36]. This grammar is designed in such a way that it can be linked with a lexicon independent of any formalism. Since our purpose in this article is to present the formal aspects of IG, we will not dwell on the experimental side.

The layout of the paper is as follows:

- Section 1 gives an intuitive view of the main IG features (polarities, superposition and underspecification) through significant examples.
- Section 2 presents the syntax of the language used to represent polarized tree descriptions, the basic objects of the formalism.
- Section 3 explains how syntactic parse trees are related to polarized tree descriptions with the notion of minimal and saturated model.
- In section 4, we illustrate the expressivity of IG with various linguistic phenomena.
- In section 5, we compare IG with the most closely related formalisms.
- Section 6 briefly presents the computational aspects of IG through their implementation in the LEOPAR parser, which works with a relatively large French interaction grammar.

[1] As no standard term exists, we use the term "superposition" to name the operation where two trees are combined by merging some nodes of the first one with nodes of the second one.

1 The main features of Interaction Grammars

The aim of this section is to give informally, through examples, an overview of the key features of IG.

1.1 A basic example

1.1.1 Syntactic tree

In IG, the parsing output of a sentence is an ordered tree where nodes represent syntactic constituents described by feature structures. An example of syntactic tree for sentence (1) is shown in Figure 1[2].

(1) *Jean la voit.*
John it sees.
'John sees it.'

Each leaf of the tree carries a phonological form which is a string that can be empty (written ϵ): in our example, *"Jean"* in node [C], *"la"* in [E], *"voit"* in [F], ϵ in [G] and *"."* in [H]). The *phonological projection* of a tree is the left to right reading of the phonological forms of its leaves (*"Jean"*$\cdot$*"la"*$\cdot$*"voit"*$\cdot\epsilon\cdot$*"."* = *"Jean la voit."* in the example).

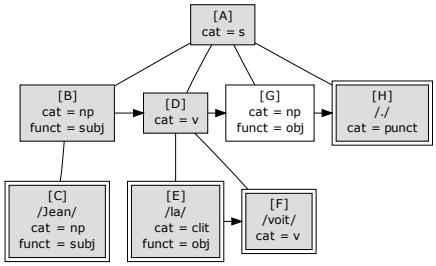
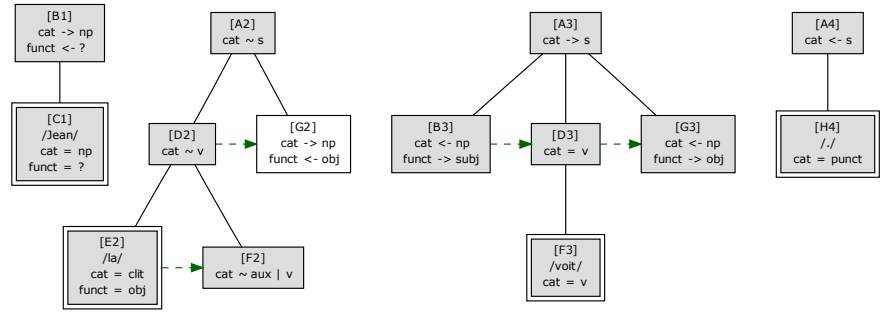


Figure 1: Syntactic tree for the sentence *"Jean la voit."*

1.1.2 Initial tree descriptions



{A2, A3, A4} $\longrightarrow$ A {B1, B3} $\longrightarrow$ B {C1} $\longrightarrow$ C {D2, D3} $\longrightarrow$ D
{E2} $\longrightarrow$ E {F2, F3} $\longrightarrow$ F {G2, G3} $\longrightarrow$ G {H4} $\longrightarrow$ H

Figure 2: IPTDs and interpretation function for the sentence *"Jean la voit."*

[2]To increase readability, only a part of the feature structures is shown in the figures; many other features (gender, number, mood, ...) are used in practice. In the following, we only show relevant features in figures.

The elementary syntactic structures are *initial polarized tree descriptions* (written IPTDs in the following). Figure 2 shows the four IPTDs used to build the syntactic tree in Figure 1. A syntactic tree is said to be a model of a set of IPTDs if each node of the syntactic tree interprets some nodes of the IPTDs and this tree satisfies saturation and minimality constraints. For our example, the interpretation function is also given in Figure 2.

IPTDs are underspecified trees: for instance, in Figure 2, the precedence relation between nodes [D2] and [G2] is large: [D2] must be to the left of [G2] but any number of intermediate nodes between [D2] and [G2] are allowed in the final tree model.

Moreover, IPTDs contain features with polarities acting as constraints. A positive (written `->`) polarity must be associated with a compatible negative (written `<-`) one: in the example, when building the model, the positive feature `cat -> s` of node [A3] is associated with the negative feature `cat <- s` of node [A4].

1.1.3 Tree descriptions

A more general notion of tree description is not strictly needed in the formalism definition, however this notion is useful to represent partial parses of sentence and to consider atomic steps in parsing process. These polarized tree descriptions (PTDs) are formally described in the next section.

1.2 Polarized features to control syntactic composition

The notion of polarity represents the core of the IG formalism.

1.2.1 Positive and negative polarities

Like in categorial grammars, resources can be identified as available (positive polarity) or needed (negative polarity). Each positive or negative feature must be neutralized by a dual polarity when the model is built. A polarity which is either positive or negative is said to be *active*.

This mechanism is intensively used. It is used similarly as in CG, for instance, to control the interactions of:

- a determiner with a noun;
- a preposition with a noun phrase;
- a verb, a predicate noun or adjective with its arguments defined in the subcategorization frame.

But polarities are also used in a more specific manner in IG to deal with other kinds of interactions. For instance:

- to handle pairs of grammatical words like *ne/pas*, ... (see below subsection 4.1);
- to manage interaction of punctuation with other constructions in the sentence;
- to link a reflexive pronoun *se* with the reflexive construction of verbs;
- to manage interaction between auxiliaries and past participles.

1.2.2 Virtual polarities

Recently, a third kind of polarity was added which is called *virtual* (written $\sim$). A feature with a virtual polarity must be combined with some other compatible feature which has a polarity different from $\sim$. It gives more flexibility to express constraints on the context in which a node can appear. Virtual polarities are used, for instance:

- to describe interaction between a modifier and the modified constituent (adverb, adjective, ...), see subsection 4.3 for an example;

- to express context constraints on nodes around the active part of a description; it allows for a control on the superposition mechanism: in Figure 2, the three nodes [A2], [D2] and [F2] with virtual `cat` polarities describe the context in which the clitic *"la"* must be used; this IPTD requires that three other non-virtual nodes compatible with [A2], [D2] and [F2] exist in some other IPTDs; in our example, non-virtual nodes [A3], [D3] and [F3] are given by the verb. This mechanism handles the constraint on the French clitic *"la"*. It comes before the verb (node [E2] before node [F2]) but contributes with an object function (node [G2] after node [F2] because the canonical position of French direct object in on the right of the verb).

1.2.3 Polarities at the feature level

A difference with respect to other formalisms using polarities is that, in IG, polarities are attached to features rather than to nodes. It is then possible to use polarities for several different features to control different types of positive/negative pairing (for instance in our grammar, the feature `mood` is polarized in the auxiliaries/past participles interaction; the feature `neg` is polarized in the interaction of the two pieces of negation).

Hence with polarities at the feature level, the same syntactic constituent can interact more than once with its environment through several feature neutralizations.

One of the typical usage of such interactions, that implies more than two nodes is subject inversion. In French, in some specific cases the subject can be put after the verb (sentences (2), (3) and (4)). However, uncontrolled subject inversion would lead to over-generation. A solution is to use two different interactions: between the subject and the verb on one hand; and on the other hand between the subject and some other word which is specific to the construction where the subject can be postponed.

(2) *Jean qu'aime Marie vient.*
John that loves Mary comes.
'John that Mary loves comes.'

(3) *Aujourd'hui commence le printemps.*
Today begins the spring.
'Today begins the spring.'

(4) *Que mange Jean ?*
What does eat John?
'What does John eat?'

In the sentence (2), the subject *"Marie"* of the verb *"aime"* can be postponed because it is in a relative clause introduced by the object relative pronoun *"que"*. Hence, in the noun phrase *"Jean qu'aime Marie"* (see figure 3), the proper noun *"Marie"* interacts both with the verb *"aime"* (neutralization of the features `cat -> np` in [A] and `cat <- np` in [B]) and with the relative pronoun *"qu' "* (neutralization of the features `funct <- ?` in [A] and `funct -> subj` in [C]). Figure 4 gives the PTD after superposition.

1.3 Tree superposition as a flexible way of realizing syntactic composition

For the grammatical formalisms that are based on trees (the most simple formalism of this type is Context Free Grammar), the mechanism of syntactic composition often reduces to substitution: a leaf L of a first tree merges with the root R of a second tree. In this way, constraints on the composition of both trees are localized at the nodes R and L. They cannot say anything about the environment of both nodes.

The TAG formalism offers a more sophisticated operation, *adjunction*, but this operation is also limited in expressing constraints on syntactic composition: instead of merging two nodes, we merge two pairs of nodes. A node N splits into an up node N_{up} and down node N_{down}, which respectively merge with the root R and the foot F of the auxiliary tree. Constraints on syntactic composition is now localized on three nodes N, R and F.

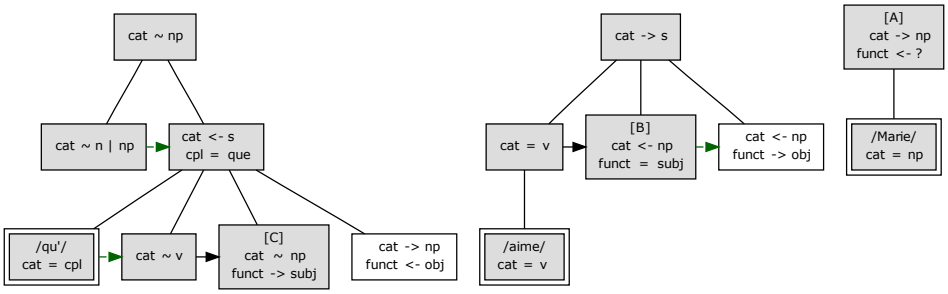


Figure 3: IPTDs for the sequence of words *"qu'aime Marie"* before superposition

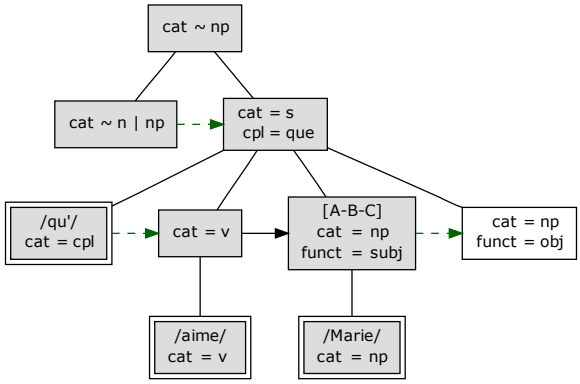


Figure 4: PTD for the sequence of words *"qu'aime Marie"* after superposition

In IG, the syntactic composition is much more flexible: we can merge any two nodes (in the same PTD or in two different ones). Then, the propagation of the constraints related to each PTD entails a partial superposition of the two tree structures around the two nodes. In this way, we can express constraints on the environment of a node.

(5) *Jean en connaît l'auteur.*
John of it knows the author.
'John knows the author of it.'

Let us consider the sentence (5). The clitic pronoun *"en"* provides the object *"auteur"* of the verb *"connaît"* with a noun complement. Our French lexicon gives the IPTD of Figure 5 to represent the syntax of this usage of the clitic pronoun *"en"*. In this IPTD, the node [N] with feature `prep -> de` represents the trace of the preposition phrase represented by the clitic *"en"* as a sub-constituent of the object of the verb. Figure 6 shows a PTD resulting from the (partial) parsing of *"connaît l'auteur"*. In this PTD, the node [M] with feature `prep <- de` represents the noun complement that is expected by the noun *"auteur"*.

Now, when we compose *"en"* with *"connaît l'auteur"* (i.e. tree descriptions of Figures 5 and 6), nodes [N] and [M] have to be merged in order to neutralize their features `cat`, `funct` and `prep`. By propagating tree well-formedness and polarity constraints, the merging of [N] and [M] entails the partial superposition (Figure 7) of the two PTDs. Note that there are 9 atomic operations of node merging during this composition.

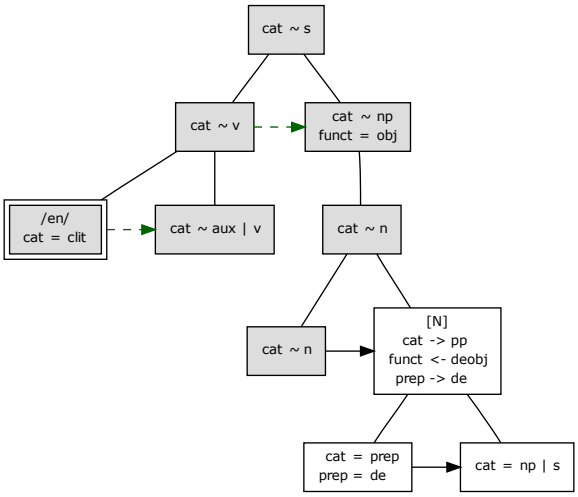


Figure 5: IPTD representing the syntax of the clitic *"en"*

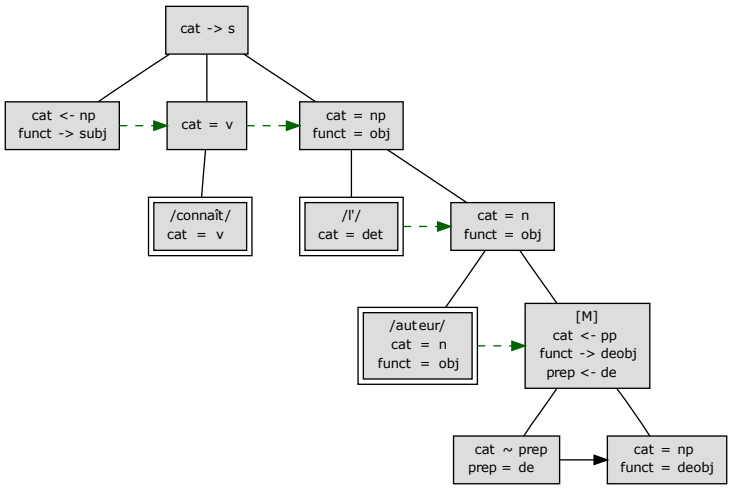


Figure 6: PTD representing the syntax of the phrase *"connaît l'auteur"*

1.4 Underspecified structures

With IG, both dominance and precedence relations can be underspecified: an IPTD can constrain a relation between two nodes without restricting the distance between the nodes in the model. Underspecified relations, combined with tree superposition, increase the flexibility of the formalism: it is possible to give more general constraints on the context of a node.

Underspecification on dominance relation makes it possible to express general properties on unbounded dependencies. For instance, the relative pronoun *"que"* can introduce an unbounded dependency between its antecedent and a verb which has this antecedent as object of adjectival complement: sentences (6) and

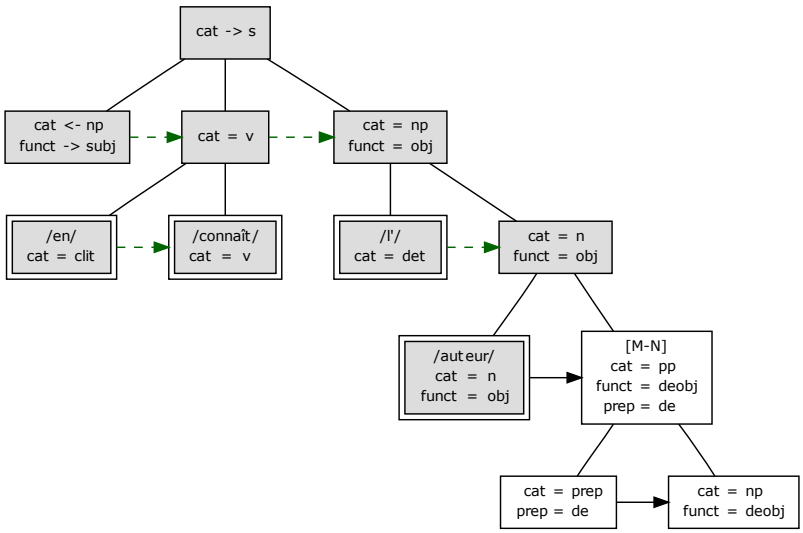


Figure 7: PTD representing the syntax of the phrase *"en connaît l'auteur"*

(7)[3].

(6) *Jean* **que** *Marie* *aime* □ *dort.*
John that Mary loves sleeps

(7) *Jean* **que** *Pierre* *croit* *que* *Marie* *aime* □ *dort.*
John that Peter thinks that Mary loves sleeps

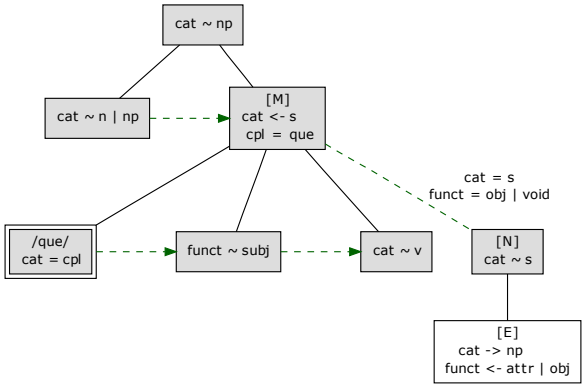


Figure 8: IPTD for the relative pronoun *que*

[3]The symbol □ indicates the original place of the extracted argument.

Figure 8 provides an IPTD to model this use of *"que"*. An empty node [E] represents the trace of an object or an adjectival phrase; [N] represents the clause in which the trace is a direct constituent and [M] represents the relative clause introduced by the relative pronoun *"que"*. [N] can be embedded at any depth in [M], which is expressed by an underspecified dominance relation. Figure 9 shows a model for the sentence (6) in which the relation is realized by merging [M] and [N], whereas Figure 10 shows a model for the sentence (7) in which the relation is realized by an immediate dominance relation.

In order to deal with island constraints, large dominances need to be controlled. In IG, this is possible with the notion of filtering feature structures. A filtering feature structure is a polarized feature structure where all polarities are neutral. A large dominance $M >^* N$ labelled with a filtering feature structure ψ means that node M must dominate N in the model and that each node along the path from M to N in this model must be compatible with ψ. For instance, in Figure 8, such a filter is used to avoid extraction through nodes that are not of category `s`.

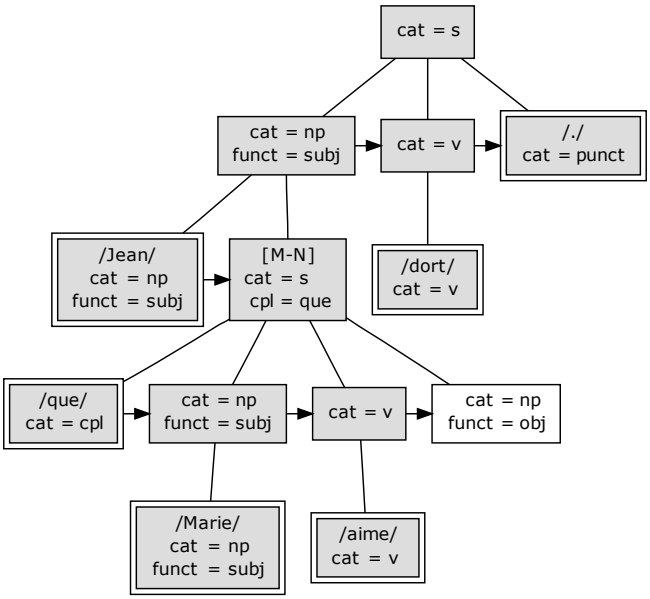


Figure 9: Syntactic tree for the sentence (6)

With underspecification on precedence relation, it is possible to describe a free ordering of some arguments. For instance, both sentences (**??**a) and (**??**b) can be parsed using the same IPTD (Figure 11) for the word *"demande"*.

(8) *Jean demande une invitation à Marie.*
John asks an invitation to Mary.
'John asks an invitation to Mary.'

(9) *Jean demande à Marie une invitation.*
John asks Mary an invitation
'John asks Mary an invitation.'

2 Formal definitions

This section is dedicated to formal definitions of IG. We define in turn:

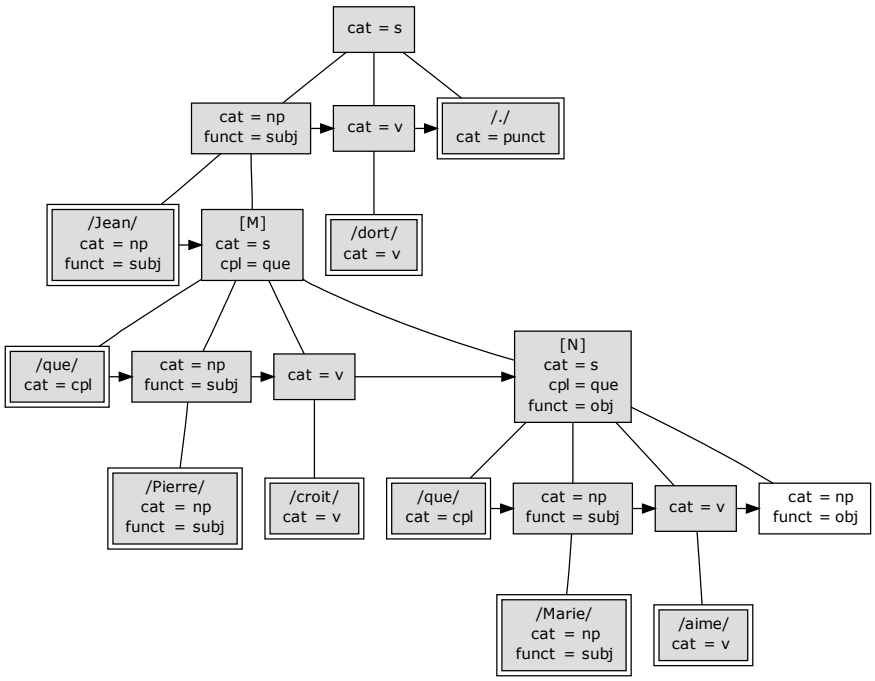


Figure 10: Syntactic tree for the sentence (7)

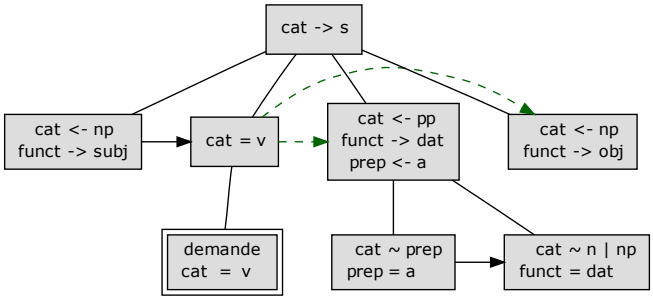


Figure 11: IPTD for the verb *"demande"*

- *syntactic trees*: the final syntactic structures in the parsing process;
- *initial polarized tree descriptions* (IPTDs): the initial syntactic structures that are associated to words at the beginning of the parsing process; PTDs are also defined as a generalization of IPTDs;
- the notion of *model* which links IPTDs and syntactic trees.

2.1 Syntactic trees

2.1.1 Features

Features are built relatively to a feature signature. A *feature signature* is defined by:

- a finite set $\mathcal{F}$ of constants called *feature names*;
- for each feature name in $\mathcal{F}$ a finite set $\mathcal{D}_f$ of constants called *atomic values*.

A *feature* is a couple (f, v) where $f \in \mathcal{F}$ and $v \in \mathcal{D}_f$ and a *feature structure* is a set of features with different feature names.

2.1.2 Syntactic trees

A *syntactic tree* is a totally ordered tree where:

- each node carries a feature structure,
- each leaf carries a string (which can be the empty string written ϵ) called *phonological form*.

In syntactic trees, parenthood relation is written $M \gg N$ (this means that M is the mother node of N), immediate precedence between sisters is written $M \lessdot\!\!\lessdot N$ (this means that M and N have the same mother and that M is just before N in the sisters ordering)[4]. We also use the notation $M \gg [N_1, \ldots, N_k]$ when the set of daughters of M is the ordered list $[N_1, \ldots, N_k]$.

Let $\gg^*$ denote the reflexive and transitive closure of $\gg$. If $M \gg^* M'$ then we call $path(M, M')$ the list of nodes from M to M':

$$path(M, M') = \{N_i\}_{1 \leq i \leq n} \text{ such that } \begin{cases} N_1 = M \\ N_i \gg N_{i+1} \text{ for } 1 \leq \text{i} < \text{n} \\ N_n = M \end{cases}$$

We define the *phonological projection* $PP(M)$ of a node M to be the list of non-empty strings built with the left to right reading of the phonological forms in the subtree rooted by M:

- if $M \gg []$ (i.e. M is a leaf) and the phonological form of M is ϵ then $PP(M) = []$,
- if $M \gg []$ and the phonological form of M is the non-empty string *phon* then $PP(M) = [phon]$,
- if $M \gg [N_1, \ldots, N_k]$ then $PP(M) = PP(N_1) \circ \ldots \circ PP(N_k)$ (where $\circ$ is the concatenation of lists).

The phonological projection of a syntactic tree is the phonological projection of its root.

We conclude here with a remark. The fact that syntactic trees are completely ordered trees can sometimes produce unwanted effects. For instance, when a node has several empty daughters, it may be not relevant to consider the relative order of these nodes. In sentences (8) and (9), the verb *"demander"* with a direct object and a dative does not impose any order between arguments. When the two arguments are realized as clitics in sentence (10), the relative order of clitics is fixed but there are two models with different ordering on empty nodes corresponding to the two arguments.

(10) *Jean la lui demande.*
John it to her asks.
'John asks it to her.'

In order to avoid this problem, it is possible to define an equivalence relation that identifies the two models of the sentence (10). We will not detail this relation in this article.

[4] We use double symbols to avoid confusion with relations that are defined later for IPTDs.

2.2 Polarized tree descriptions

2.2.1 Polarities

Polarities are heavily used in IG to take into account the resource sensitivity of natural languages. Furthermore, the parsing process strongly relies on these polarities.

The current IG formalism uses four polarities:

- *positive* (written `->`): a feature with a positive polarity describes an available resource;
- *negative* (written `<-`): a feature with a negative polarity describes a needed resource;
- *virtual* (written $\sim$): a feature with a virtual polarity is waiting for unification with another non-virtual one; virtual polarities are used for expressing constraints on the context in which an IPTD can be inserted;
- *neutral* (written `=`): a feature with a neutral polarity is not concerned by the resource management: it acts like a filter in case of unification; but unification is not required.

A multiset of polarities is said to be *globally saturated*:

- if it contains exactly one positive and one negative polarity;
- or if it contains no positive, no negative and a least one neutral polarity.

2.2.2 Polarized features

Whereas features in final syntactic trees are defined by a couple name value, in the tree description a polarity is attached to each feature and the feature values can be underspecified (with a disjunction of atomic values).

Hence, *polarized features* are now defined by triples of:

- a feature name f taken from $\mathcal{F}$,
- a polarity,
- a feature value which is a disjunction of atomic values taken from $\mathcal{D}_f$; a feature value is written as a list of atomic values separated by the pipe symbol `|`; the question mark symbol `?` denotes the disjunction of all values in $\mathcal{D}_f$.

A polarized feature is written as the concatenation of these three components (for instance `cat -> np|pp`, `funct <- ?` are polarized features).

It is also possible to give additional constraints on feature values with co-references. A co-reference is noted with `<`*i*`>`; for instance `mood = <2> ind|subj` is a co-referenced feature.

2.2.3 Polarized feature structures

A *polarized feature structure* is a set of polarized features with different feature names.

2.2.4 Filtering feature structures

Filtering feature structures are used to represent constraints on underspecified dominances. A *filtering feature structure* is a polarized feature structure where all polarities are neutral.

The constraints on underspecified dominances are stated in terms of *compatibility*. A feature structure φ is said to be *compatible* with a filtering feature structure Ψ (notation $\varphi \triangleleft \Psi$) if, for each feature name f defined in both structures, the atomic value associated with f in φ is included in the disjunction associated with f in Ψ.

2.2.5 Polarized nodes

A *polarized node* is described by:

- a polarized feature structure;
- a node type.

Node types express constraints on the phonological projection of nodes in the model. Each node has one of these four types:

- **anchor** with an associated phonological form (a non-empty string): the image of an anchor must be a leaf of the tree model (anchors are drawn with a double border in figures);
- **full**: a full node must have an image with a non-empty phonological projection;
- **empty**: an empty node must have an image with an empty phonological projection (empty nodes are drawn with white background in figures);
- **default**: a default node has no constraint on its phonological projection.

2.2.6 Polarized tree descriptions

We consider four types of relation between nodes in our tree descriptions:

dominance
The relation $M > N$ constrains the image of M to be the mother of the image of N. In such a relation it can also be imposed that N is the leftmost (resp. rightmost) daughter of M: we write $M > \bullet N$ (resp. $M > N\bullet$). Finally, an arity constraint can be expressed on the set of daughters of a node: $M > \{N_1, \ldots, N_k\}$ imposes that the image of M in the model has exactly k daughters that are images of the N_i (this arity constraint does not impose any order on the k daughters of the node M).

large dominance
$M >^* N$ constrains the image of N to be in the subtree rooted at the image of M[5]. A large dominance can also carry an additional constraint on the nodes that are on the path from M to N in the model: $M >^*_\Psi N$ (where Ψ is a filtering feature structure) constrains that the image of N is in the subtree rooted at the image of M and that each node along the path between the two images carries a feature structure which is compatible with Ψ.

precedence
$M \prec N$ constrains the images of the two nodes to be daughters of the same node in the model and the image of M to be the immediate left sister of the image of N;

large precedence
$M \prec^+ N$ constrains the images of the two nodes to be daughters of the same node in the model and the image of M to precede the image of N in the ordered tree; this precedence is strict, hence the two images have to be different.

A *polarized tree description* (PTD) is defined by:

- a set of polarized nodes;
- a set of relations on these nodes which verifies the condition: if $N_1 \prec N_2$ or $N_1 \prec^+ N_2$ then there is a node M such that $M > N_1$ and $M > N_2$.

  Note that this condition imposes that N_1 and N_2 have the same mother in the IPTD and not only in the model.

[5]Note that the symbol $>^*$ is another relation which is not defined as the reflexive and transitive closure of the relation $>$. The same remark applies to relations $\prec^+$ and $\prec$ defined below.

2.2.7 Initial polarized tree descriptions

IPTDs are elementary structures that are linked with words in the grammar; an IPTD is a PTD which verifies the additional constraint: the relation $> \cup >^*$ defines a tree structure on the nodes, this implies connexity and the fact that except the root node, all other nodes N have exactly either one mother node M ($M > N$) or one ancestor node M ($M >^* N$ or $M >^*_\Psi N$).

3 Syntactic trees as models of IPTDs

The aim of this section is to describe precisely the link between IPTDs and syntactic trees.

3.1 Syntactic trees as models of set of IPTDs

Let $\mathcal{G}$ be an interaction grammar. A syntactic tree $\mathcal{T}$ is a model of a multiset of IPTDs $\mathcal{P} = \{\mathcal{P}_i\}_{1 \leq i \leq k}$ if there is *an interpretation function* $\mathcal{I}$ from the nodes $\mathcal{NP}$ of the multiset $\mathcal{P}$ to nodes $\mathcal{NT}$ of the syntactic tree $\mathcal{T}$ such that:

Dominance adequacy

- if $M, N \in \mathcal{NP}$ and $M > N$ then $\mathcal{I}(M) \gg \mathcal{I}(N)$.

Large dominance adequacy

- if $M, N \in \mathcal{NP}$ and $M >^* N$ then $\mathcal{I}(M) \gg^* \mathcal{I}(N)$.
- if $M, N \in \mathcal{NP}$ and $M >^*_\Psi N$ then $\mathcal{I}(M) \gg^* \mathcal{I}(N)$ and for each node P in $path(\mathcal{I}(M), \mathcal{I}(N))$, $\varphi(P) \lhd \Psi$.

Precedence adequacy

- if $M, N \in \mathcal{NP}$ and $M \prec N$ then $\mathcal{I}(M) \lessdot \mathcal{I}(N)$.

Large precedence adequacy

- if $M, N \in \mathcal{NP}$ and $M \prec^+ N$ then $\mathcal{I}(M) \lessdot^+ \mathcal{I}(N)$.

Feature adequacy

- if $M \in \mathcal{NT}$ and $f = v$ is a feature of M then, for each node N in $\mathcal{I}^{-1}(M)$, either v is an admissible value for the feature f in N or N does not contain the feature name f;
- if $M, N \in \mathcal{NP}$ both contain a feature f with the same co-reference, then the values associated with f in $\mathcal{I}(M)$ and $\mathcal{I}(N)$ are identical.

Node type adequacy

- if $M \in \mathcal{NP}$ is an anchor with phonological form *phon*, then $PP(\mathcal{I}(M)) = [phon]$;
- if $M \in \mathcal{NP}$ is empty then $PP(\mathcal{I}(M)) = []$;
- if $M \in \mathcal{NP}$ is full then $PP(\mathcal{I}(M)) \neq []$.

Saturation

- the multiset of polarities associated to a feature name f in the set of nodes in $\mathcal{I}^{-1}(M)$ which contains the feature f is globally saturated.

Minimality

- $\mathcal{I}$ is surjective;
- if $M, N \in \mathcal{NT}$ and $M \gg N$ then there is $M' \in \mathcal{I}^{-1}(M)$ and $N' \in \mathcal{I}^{-1}(N)$ such that $M' > N'$;

- if $M \in \mathcal{NT}$ and $f = v$ is a feature of M then at least one node in $\mathcal{I}^{-1}(M)$ contains a feature with name f;
- if $M \in \mathcal{NP}$ is a leaf node with a non-empty phonological form *phon*, then $\mathcal{I}^{-1}(M)$ contains exactly one anchor node with phonological form *phon*.

The four points defining minimality control the fact that "nothing" is added when the model is built. They respectively control the absence of node creation, parenthood relation creation, feature creation, and phonological form creation.

Note that there can be more than one interpretation function for a given tree model.

3.2 Polarized grammars

An *interaction grammar* $\mathcal{G}$ is defined as a set of IPTDs. The tree language defined by the grammar $\mathcal{G}$ is the set of syntactic trees which are the models of a multiset of IPTDs from $\mathcal{G}$. The string language defined by a grammar is the set of phonological projections of the trees in the tree language.

We said that a syntactic tree $\mathcal{T}$ is a *parse tree* of a sentence $\mathcal{S}$, that is a list of words $\mathcal{S} = w_1, \ldots w_n$ if:

- $\mathcal{T}$ is a model of some multiset of IPTDs from $\mathcal{G}$,
- $PP(\mathcal{T}) = [w_1, \ldots, w_n]$.

An interaction grammar is said to be *lexicalized* if each IPTD contains at least one anchor (an anchor is a leaf with a non-empty phonological form).

An interaction grammar is said to be *strictly lexicalized* if each IPTD contains exactly one anchor. In this case, the link with the words of the language can be seen as a function which maps a word to the subset of IPTDs which have this word as the phonological form of its anchor. The grammar written so far for French is strictly lexicalized.

4 The expressivity of Interaction Grammars

We present four aspects of IG that highlight their expressivity. We illustrate these aspects with examples taken from our French IG because it is the only IG which is fully implemented at the moment, but there is no essential obstacle to use IG with other languages (an English IG is being written).

4.1 The use of polarities for pairing grammatical words

In French, there are some grammatical words that are used in pairs:

- comparative, *"plus ... que"* (more ... than), *"moins ... que"* (less ... than), *"si ... que"* (so ... that), *"aussi ... que"* (as ... as);
- negation, *"ne ... pas"* (not), *"ne ... rien"* (nothing), *"ne ... aucun"* (no), *"ne ... personne"* (nobody), ...;
- coordinating words like *"soit ... soit ..."* (either ... or), *"ni ... ni ..."* (neither ... nor), *"ou ... ou bien ..."* (either ... or).

The difficulty of modelling them is that their relative position in the sentence is more or less free. For instance, here are examples that illustrate various positions of the determiner *"aucun"* used with the particle *"ne"*:

(11) *[Aucun] collègue [ne] parle à la femme de Jean.*
No colleague talks to the wife of John.
'No colleague talks to John's wife.'

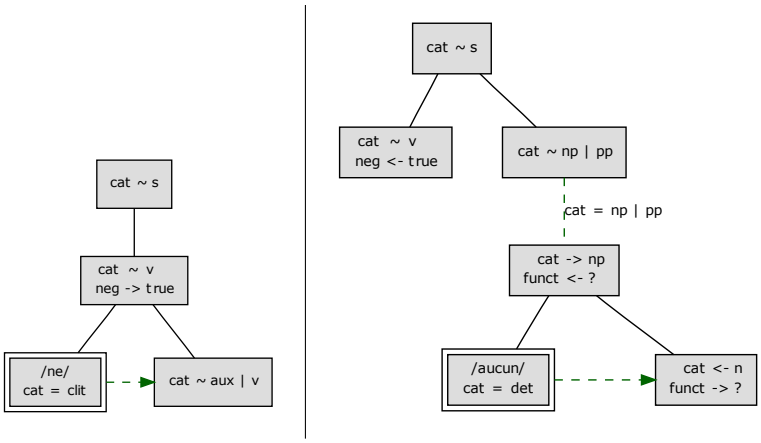


Figure 12: IPTDs associated with the particle *"ne"* and the determiner *"aucun"*

(12) *Jean [ne] parle à la femme d' [aucun] collègue.*
John talks to the wife of no colleague.
'John talks to no colleague's wife.'

(13) *Le directeur dans [aucune] entreprise [ne] décide seul.*
The director in no compagny decides alone.
'The director in no compagny decides alone.'

(14) *Jean [n'] est à la tête d' [aucune] entreprise.*
John is at the head of no compagny.
'John is at the head of no compagny.'

(15) **Jean qui dirige [aucune] entreprise, [n]'est satisfait.*
John who heads no compagny, isn't satisfied.

The IPTDs from Figure 12, associated with the words *"ne"* and *"aucun"*, allow all these sentences to be correctly parsed. The word *"ne"* put a positive feature `neg -> true` on the maximal projection of the verb that it modifies and this feature is neutralized by a dual feature `neg <- true` provided by *"aucun"*. In its IPTD, there is a constraint in the underspecified dominance relation that forbids the acceptation of the sentence (15).

4.2 Constrained dominance relations modelling long-distance dependencies

Underspecified dominance relations are used to represent unbounded dependencies and the feature structures that label these relations allow for the expression of constraints on these dependencies, such as barriers to extraction.

Relative pronouns, such as *"qui"* or *"lequel"*, give rise to unbounded dependencies in series, a phenomenon that is called *pied piping*. Sentence (16) is an example of pied piping.

(16) *Jean [dans l' entreprise de* **qui***] Marie sait que l' ingénieur travaille* □ *est malade.*
John in the compagny of whom Mary knows that the engineer works is sick.

'John, in the compagny of whom Mary knows the engineer works, is sick.'

(17)	**Jean*	*[dans*	*l'*	*entreprise*	*de*	**qui***]*	*Marie*	*qui*	*travaille*	□	*le*	*connaît*	*est*	*malade.*
	John	in	the	compagny	of	whom	Mary	who	works		knows	it	is	sick.

(18)	**Jean*	*[dans*	*l'*	*entreprise*	*qui*	*appartient*	*à*	**qui***]*	*Marie*	*travaille*	□	*est*	*malade.*
	John	in	the	compagny	which	belongs	to	whom	Mary	works		is	sick.

In example (16), there is a first unbounded dependency between the verb *"travaille"* and its extracted complement *"dans l'entreprise de qui"*. The trace of the extracted complement is denoted by the symbol □. This dependency is represented with an underspecified dominance relation in the IPTD describing the syntactic behaviour of the relative pronoun *"qui"* on figure 13. The dominance relation links the node [RelCl] representing the relative clause *"[dans l'entreprise de* **qui***] Marie sait que l'ingénieur travaille* □*"* and the node [Cl] representing the clause *"que l'ingénieur travaille* □*"*, in which the extracted prepositional phrase *"dans l'entreprise de* **qui***"* plays the role of an oblique complement. The filtering feature structure labelling the relation expresses that the path from [RelCl] to [Cl] can only cross a sequence of object clauses. This way, the sentence (17) is rejected because the dependency crosses a noun phrase, which violates the constraint.

Inside the extracted prepositional phrase, there is a second unbounded dependency between the head of the phrase and the relative pronoun *"qui"*, which can be embedded more or less deeply in the phrase. This dependency is also represented on figure 13 with an underspecified dominance relation. This dominance relation links the [ExtrPP] node and the node representing the relative pronoun *"qui"* and the associated filtering feature structure expresses that the embedded constituents are only common nouns, noun phrases or prepositional phrases. Finally, the sentence (18) is rejected.

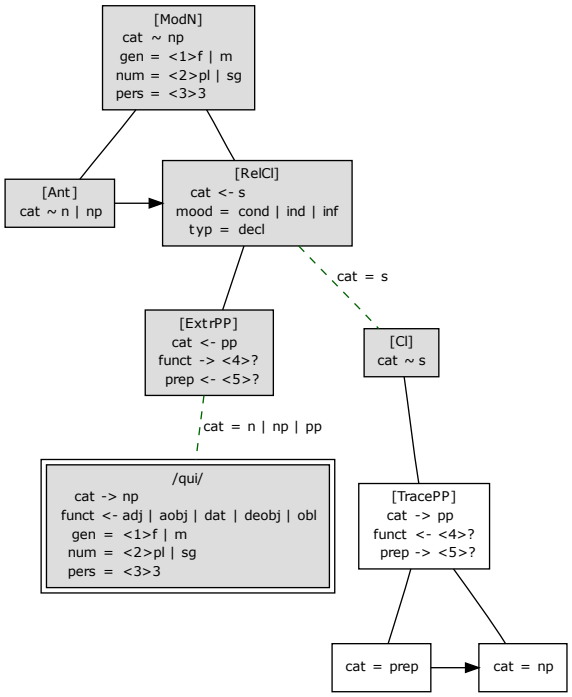


Figure 13: IPTD associated with the relative pronoun *"qui"* used in an oblique complement

4.3 Adjunction of modifiers with virtual polarities

In French, the position of adverbial complements in a sentence is relatively free, as the following examples show:

(19) ***Le soir***, *Jean va rendre visite à Marie.*
At night, John visits Mary.
'At night, John visits Mary.'

(20) *Jean,* ***le soir***, *va rendre visite à Marie.*
John, at night, visits Mary.
'At night, John visits Mary.'

(21) *Jean va rendre visite* ***le soir*** *à Marie.*
John visits at night Mary.
'John visits Mary at night.'

(22) *Jean va rendre visite à Marie* ***le soir***.
John visits Mary at night.
'John visits Mary at night.'

These variants express different communicative intentions but the adverbial complement *"le soir"* is a sentence modifier in all cases.

The virtual polarity $\sim$ was absent from the previous version of IG [35]. Modifier adjunction was performed in the same way as in several formalisms (CG, TAG) by adding a new level in the syntactic tree including the modified constituent: instead of a node with a category X, we inserted a tree with a root and two daughters;

- the root represents the constituent with the category X after modifier adjunction;
- the first daughter represents the constituent with the category X before modifier adjunction;
- the second daughter represents the modifier itself.

Sometimes, this introduction of an additional level is justified, but most of the time it brings additional artificial complexity and ambiguity. Borrowing an idea from the system of black and white polarities of A. Nasr [31], we have introduced the virtual polarity $\sim$. This polarity allows for the introduction of a modifier as an additional daughter of the node that it modifies without changing anything in the rest of the tree including the modified node. Figure 13 gives an example of an IPTD modelling a modifier: the relative pronoun *"qui"*, after combining with the relative clause that it introduces, provides a modifier of a noun phrase. The noun phrase to be modified is the antecedent of the relative pronoun, represented by node [Ant] and the noun phrase, after modification, is the root [ModN] of the IPTD.

4.4 The challenge of coordination

Even if we restrict ourselves to syntax, modelling coordination is a challenge. First, there is no consensus about the analysis of the phenomenon in thslae communauty of linguists [10, 18]. Then, whatever the chosen approach is, formalization encounters serious obstacles. In particular, both Phrase Grammars and Dependency Grammars have difficulties for modelling coordination of non-constituents.

J. Le Roux and G. Perrier propose to model coordination in IG with the notion of polarity [25, 24]. From this notion, they define the interface of a PTD as the nodes that carry positive, negative or virtual polarities. The interface characterizes the ability of a phrase to interact with other phrases. Two phrases can be coordinated if the PTDs representing their syntactic structure offer the same interface. Then, coordination consists in merging the interfaces of the two PTDs. This merging needs to superpose several positive or negative polarities and it also requires parse structure to be DAGs rather than trees. Hence, the merge

of two interfaces cannot be modelled directly in IG and it is simulated in the PTD associated with a coordination conjunction: this is divided into three parts; two lower parts are used to saturate the interfaces of the conjuncts and a higher part presents the common interface to the outside.

With this principle, it is possible to parse the following sentences, which illustrate different kinds of non-constituent coordination:

(23) *Jean [boit du vin] et [mange du pain].*
John drinks wine and eats bread.
'John drinks wine and eats bread.'

(24) *[Jean aime] mais [Marie déteste] la compétition.*
John likes but Mary dislikes competition.
'John likes but Mary dislikes competition.'

(25) *Jean donne [des fleurs à Marie] et [des bonbons à Pierre].*
John gives flowers to Mary and candies to Peter.
'John gives flowers to Mary and candies to Peter.'

(26) *La destruction [de la gare routière par les bombes] et [de la gare ferroviaire par les tanks] rend l' accès à la ville difficile.*
The destruction of the bus station by bombs and of the railway station by tanks makes access to the city difficult.
'The destruction of the bus station by bombs and of the railway station by tanks makes access to the city difficult.'

(27) *Jean voit [sa soeur lundi] et [son frère mardi].*
John sees its sister on monday and its brother on tuesday.
'John sees its sister on monday and its brother on tuesday.'

(28) *[Jean aime le ski] et [Marie □ la natation].*
John likes skiing and Mary swimming.
'John likes skiing and Mary likes swimming.'

Sentences (23) and (23) respectively illustrate left and right node raising. Sentences (25) and (26) illustrate coordination of argument clusters. Sentence (27) coordinates clusters mixing arguments and adjuncts. Sentence (28) illustrates the coordination of sentences with gaps. Here, the gap, which is represented by the □ symbol, corresponds to the elided verb *"aime"*.

5 Comparison with other formalisms

Currently, there exists no linguistic formalisms that prevails over the others. This means that the domain of natural language modelling is still in an embryonic state and the congestion of the market is not a good reason for not examining any new proposal. On the contrary, the market is open. But any new formalism has to show some advantages with respect to the established ones in order to survive. The challenge is to approximate linguistic generalities as much as possible while remaining tractable. Remaining tractable means being able to build large scale grammars and efficient parsers. Under this angle, the number of relevant formalisms is not that important: among the most well known and largely used, there are LFG, HPSG, TAG or CCG. The comparison of IG with other formalisms will highlight some of its strong features.

5.1 Categorial Grammar

The list of linguistic formalisms above mentions CCG (Combinatory Categorial Grammars) [45]. CCG are part of the CG family and since IG stems from CG, it is natural to begin the comparative study with CG.

IG shares with CG the fact that syntactic composition is based on the resource sensitivity of natural languages, a property which is built-in in both kinds of formalisms. However, they differ in the framework that they use. For this, we refer again to the distinction between two approaches for syntax introduced by G. Pullum and B. Scholtz [37] and we can claim that CG uses a generative-enumerative syntactic (GES) framework whereas IG uses a model-theoretic syntactic (MTS) framework. In other words, CG derives all acceptable sentences of a language from a finite set of axioms, the lexicon, using a finite set of rewriting rules. IG associates sentences with a set of constraints, which are solved to produce their syntactic structures.

[34] proposes a method for transposing grammars from the GES to the MTS framework under some conditions. This method applies to CG and can be used to compare IG with CG by putting them in the same MTS framework. The precise description of such a translation goes beyond the goal of this article but we give an outline of its output.

To be more precise, let us focus on a particularly interesting member of the CG family: CCG. The formalism of CCG is a very good compromise between expressivity, simplicity and efficiency. At the same time, it is able to model difficult linguistic phenomena, the most famous being coordination [44], and it is used for parsing large corpora with efficient polynomial algorithms and large scale grammars [19, 11].

If we use the method proposed in [34] to translate a particular CCG in the MTS framework, we obtain a very specific IG with the following features as output:

- Each syntactic type is translated into an IPTD with a particular shape. Nodes are labelled with feature structures which contains only the `cat` feature. The values of this feature are the atomic types of the CCG. Immediate dominance relations always go from nodes with a positive feature to nodes with a negative feature (possibly with intermediate nodes without labels). For large dominance relations, this is the contrary.

- In the output IPTD, there are no precedence relations. Word order is controlled by a special feature `phon`, which gives the phonological form of each node. This feature is neutral and takes its values from the monoid of the words of the language. We need to extend the system of IG feature values to allow the presence of variables inside terms representing `phon` values. These variables are used to model the sharing of unknown substrings of words by `phon` values of different nodes.

- Successful CCG derivations are translated into constructions of IPTD models. However, all valid IPTD models do not correspond to successful derivations, because the particular form of the combinatory rules imposes constraints to superposition. Conversely, in very rare cases, CCG derivations cannot be translated into constructions of IPTD models because of two rules: backward and forward crossed compositions. By allowing word permutation, these rules contradict the monotony of the MTS framework. A simple solution consists in discarding the two problematic rules and considering only a restriction of CCG.

Even if the translation of a CCG into an IG is not perfect, this highlights the difference between the two formalisms. CCG can be viewed as IG with additional constraints on the form of IPTDs and superpositions. What is important, is that node merging is restricted to pairs of nodes with dual `cat` features. This has two important consequences:

- It is not possible to express passive constraints on the environment of a syntactic object, as we do in IG using nodes with virtual and neutral features.

- The internal structure of an IPTD, that is its saturated nodes, is ignored by CCG. The only thing that matters is its interface, that is its unsaturated nodes.

The abstraction power that is expressed by this last remark is a source of over-generation for CCG. To limit over-generation, [3] have introduced modalities to control the applicability of combinators rules. These

modalities are specified in the lexicon, so that the syntactic behaviour of a word can be more or less constrained. The problem is that we cannot relativize these constraints with respect to the environment in which the word can be situated. For instance, consider the following sentence:

(29) Mary whom John met yesterday is my wife.

In CCG, the relative pronoun *"whom"* provides an object for the clause that it introduces on the right periphery of this clause, but the transitive verb *"met"* expects its object immediately on its right. The way to solve this contradiction is to assign a modality to the lexical entries of *"met"* and *"yesterday"*, which allow the permutation of the object of *"met"* with *"yesterday"*. But, doing this, we make the following sentence acceptable:

(30) * John met yesterday Mary.

IG does not present such an drawback, because *"yesterday"* is taken as a sentence modifier and it is modelled according to the method presented in subsection 4.3.

To summarize, multi-modal CCG limits over-generation but does not eliminate it.

5.2 Dependency Grammars

Like CG, Dependency Grammar (DG) [32] does not denote a unique formalism but rather a family of formalisms. At the root of this family, there is the concept of *dependency*. A dependency links two words in an asymmetrical manner: one word is the *régissant* and the second word is the *subordonné*, according to the terminology introduced by L. Tesnière, the pioneer of DG [47].

Even if there is no explicit notion of polarity in DG, this underlies the notion of dependency. The potentiality of two words to establish a dependency between themselves can be expressed by equipping the *régissant* with a negative feature and the *subordonné* with a positive feature, the two features having the same value, the POS (part-of-speech) of the *subordonné* for instance. This is the general idea, which must be made more precise by examining the different DG formalisms. A key feature which differentiates DG variants is the relationship between dependency structure and word order.

Projective DG forbid cross-dependencies. They have interesting computational properties and they can be easily translated into phrase structure grammars, especially Adjukiewicz-Bar-Hillel (AB) grammars [4]. Since AB grammars can be viewed as CCG with only two combinatory rules, forward and backward applications, the consequence is that projective DG can be translated into IG following the method presented above. This translation highlights the limits of projective DG. In fact, these are not expressive enough to represent cross-dependencies or long-distance dependencies.

If we look at non projective DG, there is no formalism that has reached sufficient maturity to be used for developing real grammars. Nevertheless some works are promising and we propose to focus on Generalized Categorial Dependency Grammar (GCDG) [13], which constitute a good compromise between expressivity and complexity.

GCDG include two kinds of dependencies, thus giving birth to two independent formal systems:

- projective dependencies are represented by AB grammars, slightly extended to better take modifiers into account,
- discontinuous dependencies are represented with polarities that neutralize themselves in dual pairs.

A word that is able to govern another one in a discontinuous dependency is equipped with a negative polarity typed by the category of the *subordonné* and the *subordonné* is equipped with the dual polarity.

This representation of discontinuous dependencies makes the comparison with IG difficult. It is not possible to translate it in the framework of IG because it has no simple relationship with dominance and precedence relations, which consider phrases and not words. In IG, discontinuous dependencies are generally represented in the IPTD associated with only one of the word responsible for the dependency, by means of an underspecified dominance relation (see section 4).

Another reason that makes the comparison between IG and GCDG difficult is that there is no effective GCDG for any language. Nevertheless, we can make some remarks. In GCDG, the iteration operator $*$ allows to represent modifiers by sister adjunction as in IG. On the other hand, the hermetic separation between the two kinds of dependencies does not allow to express that the same words require a dependency when it does not matter if the dependency is projective or discontinuous.

Because of the fine dependency structure that they propose, GCDG can contribute to make clearer a controversial issue in DG, the analysis of grammatical function words, but they will be confronted to syntactic constructions, which remain problematic for all DG: coordination for instance.

5.3 Unification Grammars

The family of Unification Grammars (UG) includes all formalisms for which the mechanism of unification between feature structures occupies a central position. HPSG [41] is the member of this family for which the idea is integrated as completely as possible. The grammatical objects are typed feature structures (grammatical rules, lexical entries and partial analysis structures) and the only composition operation is unification.

From some angle, HPSG feature structures can be viewed as DAGs, in which edges are labeled with feature names and leaves with atomic feature values. In this way, unification appears as DAG superposition. As in IG, superposition gives flexibility to HPSG and allows to represent sophisticated passive contexts of syntactic constructions.

The main difference is that the notion of unsaturated structure is not built-in in the composition mechanism such as for IG with the notion of polarity. However, this notion is present in some grammatical principles such as the *Valence Principle*.

Moreover, HPSG presents three important differences with respect to IG

- DAG are more expressive than trees. In this way, some phenomena are easier to model with HPSG than with IG. For instance, factorization, which is specific to coordination, is directly represented in HPSG [29], whereas it must be simulated in IG (see paragraph 4.4 and [25]).

- Underspecification is more restricted in HPSG than in IG; it reduces to the underspecification associated with unification. All dominance relations are completely specified, so that unbounded dependencies are represented with another mechanism: the *slash feature*, the propagation of which allows to mimic unbounded dependencies.

- word order is not expressed by linear order between DAG nodes but with a specific feature PHON.

Lexical Functional Grammar (LFG) [9] is another well known member of the UG family, but because of their functional structures paired with constituency structures, they are difficult to compare with IG IPTDs. Nevertheless, presenting functional structures as path equations allows the expression of a form of underspecification, which is not present in HPSG but which exists in IG: the concept of *functional uncertainty* is similar to the IG notion of large dominance, with the same possibility of constraining the dominance path between nodes without determining its length.

Tree Adjoining Grammar (TAG) [1] is often ranked in the UG family, even if they are rather tree grammars but their use of unification is more limited: contrarily to previous formalisms, it cannot be used to superpose structures. Structures only combine by adjunction, which greatly limits the expressivity of the formalism.

6 Computational aspects

A question that arises naturally for a new formalism is its complexity. The theoretical complexity is an important point but the less formal notion of "practical" complexity is also crucial for applications. The practical complexity can be thought as: "how does the formalism behave with real grammars and real sentences?".

It is clear that IG is not as mature as the other formalisms presented in the previous section. However, some theoretical and practical works presented in this section give some insights about this question in the IG framework.

The current work focuses on strictly lexicalized IG: the methods and algorithms presented in this section apply to grammars where each IPTD contains exactly one anchor. For such a grammar, we call *lexicon* the function that maps each word to its corresponding set of IPTDs. However, it is easy to transform any lexicalized grammar into an equivalent strictly lexicalized grammar with the mechanism used in section 4.1.

In the particular case of strictly lexicalized grammar, the definition of section 3.2 can be refomulated as follows. A sentence $\mathcal{S} = w_1, \dots w_n$ has a parse tree $\mathcal{T}$ iff there is an ordered list of IPTDs $\mathcal{P} = [\mathcal{P}_1, \dots, \mathcal{P}_n]$ such that:

- for all $1 \leq i \leq n$, w_i is the phonological form of the anchor of $\mathcal{P}_i$;
- $\mathcal{T}$ is a model of the multiset $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$;
- $PP(\mathcal{T}) = [w_1, \dots, w_n]$.

Hence, the parsing process can be divided in two steps: first, select for each word of the sentence one of the IPTDs given by the lexicon; then build a syntactic tree which is a model of the list of IPTDs chosen in the first step. The choice of one IPTD for each word of the sentence is called a *lexical selection*.

6.1 Complexity

The general parsing problem for IG is NP-complete, even if the grammar is strictly lexicalized. It can be shown for instance with an encoding of a fragment of linear logic (Intuitionistic Implicative Linear Logic) in IG. Intuitively, the complexity has two sources:

- **Lexical ambiguity**. In a lexicalized IG, each word of the lexicon can be associated to several IPTDs. Hence, the numbers of lexical selections for a given sentence grows exponentially with the number of words it contains.
- **Parsing ambiguity**. When a lexical selection is done, a model should be built for the corresponding list of IPTDs. Building a model is equivalent to finding a partition on the set of nodes of the IPTDs such that each node obtained by the mergings of nodes that are in the same subset of the partition are saturated. Once again, there is an exponential number of possible partitions.

The next two subsections address these two sources of ambiguity. As already mention above, we address the problem of practical complexity. Hence, we are looking for algorithms which behave in an interesting way for real NLP grammars. For instance, the formalism can be used to define a grammar without any active polarity, but this is clearly out of the IG "spirit". The methods described below are designed for well-polarized grammars.

6.2 Global filtering of lexical selections

In this section, we describe a method which is formalized in a previous paper [6] and we see how it applies to the IG formalism. The idea is close to tagging, but it relies on more precise syntactic descriptions than POS-tagging. Such methods are sometimes called super-tagging [8]: we consider an abstraction of our syntactic structures for which parsing is very efficient even if this abstraction brings over-generation. The key point is that a lexical selection which is not parsed in the abstract level cannot be parsed in the former level and can be safely removed.

In IG, we consider as an abstract view of an IPTD the multiset of active features present in the IPTD. Then, a lexical selection is valid in the abstract level if the union of the multiset associated to IPTDs is globally saturated.

The parsing at this abstract level is efficient because it can be done using finite state automata (FSA). For each couple (f, v) of a feature name and a feature value, an acyclic automaton is build with IPTDs as edges and integers as state: the integer in a state is the count of polarities (positive counts for 1 and negative for -1) for the couple (f, v) along every paths from initial state to the current state. Finally, only lexical selections which end with a state labelled with 0 should be kept.

An automaton is built for each possible couple (f, v), then a FSA intersection of the set of automata describes the set of lexical selections that are globally saturated.

The fact that feature values can be disjunction of atomic values in IPTDs causes the automata to be non deterministic. We turn them into deterministic ones using intervals of integers instead of integers in states of the automaton.

When a grammar uses many polarized features, the method can be very efficient and remove many bad lexical selections before the deep parsing step. For instance, for the sentence (6.2) the number of lexical selections reduces from 578 340 to 115 (in 0.08s).

(31) *L'* *ingénieur* *le* *présente* *à* *l'* *entreprise.*
The engineer him presents to the enterprise.
'The engineer presents him to the enterprise.'

The main drawback of the method is that the count of polarities is global and does not depend on word order: any permutation of a saturated lexical selection is still saturated. Some recent or ongoing works try to apply some finer filters on automaton. In [7], a specialized filter is described dealing with coordination for instance. For each IPTD for a symmetrical coordination, this filter removes the IPTD if it is not possible to find two sequences of IPTD on each side of the coordination with the expected multiset of polarities.

6.3 Deep parsing

Deep parsing in IG is a constraint satisfaction problem. Given a list of IPTDs, we have to find the set of models of the corresponding multiset which respects the word order of the input sentence.

Three algorithms have been developed for deep parsing in IG:

Incremental This algorithm scans the sentence word by word. An atomic step consists in chosing a couple of positive and negative features to superpose. In others words, an interpretation function is built step by step, guided by the **saturation** property of models.

CKY-like The CKY-like algorithm, as the incremental one, tries to build the interpretation function step by step. The difference with the previous one is the way the sentence is scanned; it is done by filling a chart with partial parsing corresponding to sequence $[i, j]$ of consecutive words.

Earley-like This last algorithm tries to build at the same time the tree model and the interpretation function. It proceeds with a top-down/left-right building of the tree.

6.3.1 Node merging

The first two algorithms use the same atomic operation of node merging. This operation takes as input a PTD D and a couple of nodes (N_1, N_2); it returns a new PTD D' which verifies that each model of D' is a model of D.

The model searching can be decomposed in small node merging steps because of the following property: if the unsaturated PTD D has a model T then there are two dual nodes N_1 and N_2 such that T is still a model of the PTD obtained by merging of N_1 and N_2 in D.

Technically, when two nodes are merged, some other constraint propagation rules can be applied to the output description without changing the set of models. For instance, if $M_1 > N_1$ and $M_2 > N_2$ and N_1 is merged with N_2 then M_1 is necessarily merged with M_2.

6.3.2 The incremental algorithm

As already said, there is an exponential number of possible choices of couples of nodes to merge. The incremental algorithm tries to mimic the human reading of a sentence and uses a notion of bound inspired by psycholinguistics motivations to guide the parsing. This notion of bound is used in a very similar spirit in Morrill's works [28].

The psycholinguistic hypothesis is that the reading uses only a small memory to represent the already read part of the sentence. Hence, we bound the number of unresolved dependencies that can be left open while scanning the sentence. In our context, we bound the number of active polarities. Then the algorithm uses a kind of shift/reduce mechanism: we start with an empty PTD and then we used recursively the two rules:

REDUCE if the current PTD has a number of active polarities greater than the bound or if there is no more IPTD to add, then try the different ways to neutralize two dual active features;

SHIFT else, add the next IPTD to the current PTD.

In the LEOPAR implementation, the search space is controlled in the REDUCE operation. Couples of active polarities are ordered in such a way that multiple constructions of the same model which differ only by a permutation on the neutralizations order are avoided.

6.3.3 The CKY-like algorithm

The well-known CKY parsing algorithm for CFG can be adapted to IG. The basic idea is to focus on contiguous sequence of words and to use the following informal rule:

A PTD for a sequence $[i, j]$ is obtained with a neutralization of two dual features in two different PTDs for sequences $[i, k]$ and $[k + 1, j]$.

This rule is used recursively to fill a chart. In the end, we consider the PTDs obtained for the whole sentence and search for models: use the REDUCE rule of the previous algorithm until there is no more active polarity and second, build a totally ordered tree which is a model of the saturated PTD obtained in the first step.

The advantages of this algorithm is that it does not depend on a bound and that it is able to share more sub-parsing. The drawback is that it is designed to find only models that follow some continuity conditions: for instance, it is not able to find a model if neutralization arises between w_1 and w_3 in a 3 words sentence. However, in our French grammar, this condition is most of the time respected. But this algorithm should be generalized in order to deal with other languages.

6.3.4 The Earley-like algorithm

Another algorithm inspired by the classical Earley parsing algorithm for CFG has been developed for IG. The algorithm is described in [24, 26]. It is being implemented in LEOPAR and the current version is not very efficient but we hope to improve it for the next release.

There are two main difficulties to adapt this classical algorithm to IG. First, when trying to build the tree model top-down, we have to deal with large dominance relations. If the node M is used to build a node in the tree model and if $M >^* N$, then the node N must be used at any depth in the construction of the sub-tree rooted in M. Our solution is to include in each item a set of nodes that must be used in the subtree rooted at the current node. The other difficulty is to deal with the fact that the daughters of a node are only partially ordered in IPTDs and that we have to consider every total ordering compatible with the partial order when building the tree structure of the model.

6.4 Implementation

The IG formalism is implemented in a parser named LEOPAR. This software contains several modules which are used in turn for sentence parsing.

Tokenizer a minimal tokenizer is included: it allows to deal with usual tokenization problems like contraction (for instance in French, the written word *"au"* should be understood as the contraction of the two words *"à"* and *"le"*). The tokenizer returns an acyclic graph to represent tokenization ambiguities.

Lexer a flexible system of linguistic resources description is used in LEOPAR. Several levels of description can be used to described various linguistic information: morphological, syntactical,.... Unanchored IPTDs are read in an XML format produced by XMG [14] (an external tool which provides a high level language to build large coverage grammars). The anchoring mechanism is controlled by the notion of interface:

- each description tree of the unanchored grammar is associated with a feature structure called interface;
- each word is linked to a set of usages: a usage is a feature structure which describes the morphological and syntactical properties of a word;
- if an interface $I(T)$ of a tree description T unifies with a word usage U associated with a word w: then an IPTD T' is produced from T with w as phonological form.

The lexer outputs an acyclic graph which edges are labelled by IPTDs.

Filter this stage implements the global filtering of lexical selections presented above (subsection 6.2). It takes as input the acyclic graph given by the lexer and returns another acyclic graph which paths are the lexical selections kept by the filtering process.

Deep parser the final stage is the building of a set of models for the acyclic graph given by the previous stage. Implemented algorithms are adapted to deal with the sharing given by the graph representation of the ambiguity in the output of the filtering process.

The whole system can be used either with commands or through an interface. In the interface, an interactive mode is available. The user can choose a path in the automaton given by the filter stage and then choose couple of nodes to merge: this interactive mode is very useful in grammar testing/debugging.

7 Conclusion

In this paper, we focused on a formal presentation of IG, highlighting their originality with their ability to express various and sophisticated linguistic phenomena. We left both Language-Theoretic properties and implementation aspects of IG aside, as they need to be studied for themselves.

One of our fundamental ideas is to combine theory and practice. The formalism of IG is implemented in the LEOPAR parser in the same form as it is described in this paper. In this way, it can be validated experimentally. To use LEOPAR on large corpora, we need resources. There exists a French IG with a relatively large coverage [36], which is usable with a lexicon independent of the IG formalism [17]. There exists a lexicon with a large coverage available in the format required by the grammar: the Le*fff* [42]. The Le*fff* contains about 500 000 inflected forms corresponding, among others, to 6 800 verb lemmas, 37 600 nominal lemmas and 10 000 adjectival lemmas. With the Le*fff* and the French IG, LEOPAR is on the way of parsing real corpora.

The formalism is not definitively fixed and the forward and backward motion between theory and practice is important to improve it step by step. Among the questions to be studied in a deeper way, there are:

the form of the syntactic structure of a sentence: phenomena such as coordination or dislocation show that the notion of syntactic tree is too limited to express the complexity of the syntactic structure of sentences; structures as directed acyclic graphs fit in better with these phenomena;

the enrichment of the feature dependencies: dependencies between features are frequent in linguistic constructions but they cannot be represented in a compact way in the current version of IG; all cases

have to be enumerated, which is very costly; it seems not to be a difficult problem to enrich the feature system in order to integrate these dependencies.

The paper is restricted to the syntactic level of natural languages but syntax cannot be modelled without any idea of the semantic level and of the interface between the two levels; [35] presents a first proposal for the extension of IG to the semantic level but we can envisage other approaches using existing semantic formalisms such as MRS [12] or CLLS [16].

References

[1] A. Abeillé and O. Rambow, editors. *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. Stanford, CSLI, 2001.

[2] K. Adjukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935.

[3] J. Baldridge and G.-J. Kruijff. Multi-Modal Combinatory Categorial Grammar. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL '2003)*, Budapest, Hungary, 2003.

[4] Y. Bar-Hillel, H. Gaifman, and E. Shamir. On categorial and phrase structure grammars. *Bulletin of the research council of Israel*, 9F:1–16, 1960.

[5] G. Bonfante, B. Guillaume, and G. Perrier. Analyse syntaxique électrostatique. *Traitement Automatique des Langues*, 44(3):93–120, 2003.

[6] G. Bonfante, B. Guillaume, and G. Perrier. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *20th International Conference on Computational Linguistics, CoLing 2004, Genève, Switzerland*, pages 303–309, 2004.

[7] G. Bonfante, J. Le Roux, and G. Perrier. Lexical Disambiguation with Polarities and Automata. In *Lecture Notes in Computer Science*, volume 4094, pages 283–284. Springer-Verlag, 2006.

[8] P. Boullier. Supertagging: a non-statistical parsing-based approach. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 55–65, Nancy, France, April 2003.

[9] J. Bresnan. *Lexical-Functional Syntax*. Blackwell Publishers, Oxford, 2001.

[10] R. Carston and D. Blakemore. Introduction to coordination: Syntax, semantics and pragmatics. *Lingua*, 115(4):353–358, 2003.

[11] S. Clark and J.R. Curran. Parsing the wsj using ccg and log-linear models. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, pages 103–110, Barcelona, Spain, 2004.

[12] A. Copestake, D. Flickinger, K. Pollard, and I.A. Sag. Minimal Recursion Semantics - an Introduction. *Research on Language and Computation*, 3:281–332, 2005.

[13] M. Dekhtyar and A. Dikovsky. Generalized Categorial Dependency Grammars. In Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, editors, *Pillars of Computer Science*, volume 4800 of *Lecture Notes in Computer Science*. Springer, 2008.

[14] D Duchier, J. Le Roux, and Y. Parmentier. The Metagrammar Compiler: an NLP Application with a Multi-paradigm Architecture. In *Second International Mozart/Oz Conference, MOZ 2004, Charleroi, Belgium*, pages 175–187, 2004.

[15] D. Duchier and S. Thater. Parsing with tree descriptions: a constraint based approach. In *Natural Language Understanding and Logic Programming NLULP'99,Dec 1999, Las Cruces, New Mexico*, 1999.

[16] M. Egg, A. Koller, and J. Niehren. The Constraint Language for Lambda Structures. *JOLLI*, 10:457–485, 2001.

[17] C. Gardent, B. Guillaume, G. Perrier, and I. Falk. Extracting subcategorisation information from maurice gross' grammar lexicon. *Archives of Control Sciences*, pages 289–300, 2005.

[18] D. Godard. Problèmes syntaxiques de la coordination et propositions récentes dans les grammaires syntagmatiques. *Langages*, 160:3–24, 2005.

[19] J. Hockenmaier. Parsing with generative models of predicate-argument structure. In *41st Annual Meeting of the Association for Computational Linguistics (ACL '03)*, pages 359–366, Sapporo, Japan, 2003.

[20] O. Jespersen. *Analytic Syntax*. Allen and Unwin, London, 1937.

[21] S. Kahane. Polarized unification grammars. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 137–144, Sydney, Australia, 2006.

[22] L. Kallmeyer. *Tree Description Grammars and Underspecified Representations*. PhD thesis, Universität Tübingen, 1999.

[23] J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–169, 1958.

[24] J. Le Roux. *La coordination dans les grammaires d'interaction*. PhD thesis, Université Nancy 2, 2007.

[25] J. Le Roux and G. Perrier. La coordination dans les grammaires d'interaction. *Traitement Automatique des Langues*, 47(3):89–113, 2006.

[26] J. Marchand. Algorithme de Earley pour les grammaires d'interaction. Travaux universitaires, Université Nancy 2, 2006.

[27] M. Marcus, D. Hindle, and M. Fleck. D-Theory: Talking about Talking about Trees. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 129–136, 1983.

[28] G. Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319 – 338, 2000.

[29] F. Mouret. *Grammaire des constructions coordonnées. Coordinations simples et coordinations à redoublement en français contemporain*. PhD thesis, Université Paris 7, 2007.

[30] R. Muskens and E. Krahmer. Talking about trees and truth-conditions. In *Logical Aspects of Computational Linguistics, LACL'98. Grenoble, France*, 1998.

[31] A. Nasr. A formalism and a parser for lexicalised dependency grammars. In *4th Int. Workshop on Parsing Technologies, Prague, Czechoslowakia*, pages 186–195. State University of NY Press, 1995.

[32] J. Nivre. Dependency Grammar and Dependency Parsing. MSI report 04071, Växjö University: School of Mathematics and Systems Engineering., 2005.

[33] G. Perrier. Interaction grammars. In *18th International Conference on Computational Linguistics, CoLing 2000, Sarrebrücken*, pages 600–606, 2000.

[34] G. Perrier. Intuitionistic multiplicative proof nets as models of directed acyclic graph descriptions. In *8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning - LPAR 2001, 2001*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 233–248, Havana, Cuba, 2001.

[35] G. Perrier. La sémantique dans les grammaires d'interaction. *Traitement Automatique des Langues (TAL)*, 45(3):123–144, 2005.

[36] G. Perrier. A French Interaction Grammar. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, Nicolas Nicolov, and Kiril Simov, editors, *RANLP 2007*, pages 463–467, Borovets Bulgarie, 2007. IPP & BAS & ACL-Bulgaria.

[37] G.K. Pullum and B.C. Scholz. On the Distinction between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. In P. de Groote, G. Morrill, and C. Retoré, editors, *Logical Aspects of Computational Linguistics, LACL 2001, Le Croisic, France*, volume 2099 of *Lecture Notes in Computer Science*, pages 17–43. Springer Verlag, 2001.

[38] O. Rambow, K. Vijay-Shanker, and D. Weir. D-tree substitution grammars. *Computational Linguistics*, 27(1):87–121, 2001.

[39] C. Retoré. The Logic of Categorial Grammars, 2000. avalaible at URL: http://www.cs.bham.ac.uk/research/conferences/esslli/.

[40] J. Rogers and K. Vijay-Shanker. Reasoning with descriptions of trees. In *30th Annual Meeting of the Association for Computational Linguistics*, pages 72–80, 1992.

[41] I.A. Sag, T. Wasow, and E.M. Bender. *Syntactic Theory : a Formal Introduction*. Center for the Study of Language and INF, 2003.

[42] B. Sagot, L. Clément, E. de La Clergerie, and P. Boullier. The lefff 2 syntactic lexicon for french: architecture, acquisition, use. In *LREC 06, Genova, Italy*, 2006.

[43] E. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics, LACL'96, Nancy, France*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95, 1997.

[44] M. Steedman. Dependency and Coordination in the Grammar of Dutch and English. *Language*, 61(3):523–568, 1985.

[45] M. Steedman. *The Syntactic Process*. Bradford Books. MIT Press, 2000.

[46] L. Tesnière. Comment construire une syntaxe. *Bulletin de la Faculté des Lettres de Strasbourg*, 7 - $12^{ième}$ année:219–229, 1934.

[47] L. Tesnière. *Eléments de syntaxe structurale*. Librairie C. Klincksieck, Paris, 1959.

[48] K. Vijay-Shanker. Using description of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517, 1992.