Res on Lang and Comput (2009) 7:171–208
DOI 10.1007/s11168-010-9066-x

Interaction Grammars

Bruno Guillaume · Guy Perrier

Published online: 6 February 2010

Abstract Interaction Grammars are 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 · Polarity · Tree description

0 Introduction

Interaction Grammars (IGs) are a grammatical formalism based on an old idea of Jespersen (1937), Tesnière (1934) and Adjukiewicz (1935): 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 (CGs) (Carpenter 1998; Retoré 2005). In CGs, constituents are equipped with types, which express their interaction ability in terms of syntactic categories. A way

B. Guillaume (✉)
LORIA, INRIA, Nancy, France
e-mail: Bruno.Guillaume@loria.fr

G. Perrier
LORIA, Université Nancy2, Nancy, France
e-mail: Guy.Perrier@loria.fr

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 trees in which all polarities have been saturated.

The notion of polarity in this sense was not used explicitly in computational linguistics until recently. To our knowledge, Nasr (1995) was the first to propose a formalism using polarized structures. Then, nearly at the same time, Muskens and Krahmer (1998), Duchier and Thater (1999), and Perrier (2000) proposed grammatical formalisms using polarities. The latter was a first version of IGs, 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 (Perrier 2005). Then, Kahane (2006) showed that many well-known formalisms (CFG, TAG, HPSG, LFG) can be viewed as polarized formalisms. Unlike the previous approaches, polarities are used in a non monotonous way in Minimalist Grammar (MG). Stabler (1997) 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 IGs, we highlighted the fundamental mechanism of saturation between polarities underlying CGs in a more refined way, because polarities are attached to the features used to describe constituents and not to the constituents themselves. Moreover, the system of polarities is enriched with the addition of virtual polarities, which implies a generalization of the operation of neutralization into an operation of saturation.

However, the essential difference lies in the change of framework: CGs are usually formalized in a generative deductive framework, the heart of which is the Lambek Calculus (Lambek 1958), whereas IGs are 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. Pullum and Scholz (2001) highlighted the advantages of this change of framework. 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: 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 Marcus et al. (1983) to reduce non-determinism in the parsing of natural languages. It was used again by Vijay-Shanker (1992) to represent the adjoining operation of TAG in a monotonous way. Then, it was studied systematically from a mathematical point of view (Rogers and Vijay-Shanker 1992) and it gave rise to new grammatical formalisms: Tree Description Grammars (Kallmeyer 1999), D-Tree Substitution Grammars (Rambow et al. 2001).

If model theory provides a declarative framework for IGs, polarities provide a step by step operational method to build models of tree descriptions. A tree description can be viewed as a set of partially specified trees, which 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 IGs 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 IGs. The reader can find a first proposal in Perrier (2005).

An important concern with IGs is to provide a realistic formalism, which can be tested experimentally by parsing actual corpora. In order to combine the theoretical development of the formalism with experimentation, we have designed a parser based on IGs (Bonfante et al. 2003) and named LEOPAR (freely available at http://leopar.loria.fr/). 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 (Duchier et al. 2004), we have built a French interaction grammar with a relatively large coverage (Perrier 2007). 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 IGs, 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 formal material used to define syntactic trees and 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 Sect. 4, we illustrate the expressivity of IGs with various linguistic phenomena.
- In Sect. 5, we compare IGs with the most closely related formalisms.
- Section 6 shows that the concept of polarity, which grounds the formalism of IGs, allows original methods of parsing. These methods are briefly presented through their implementation in our parser, which works with a relatively large French interaction grammar.

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 IGs. The formal definitions are given in the next section.

[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.

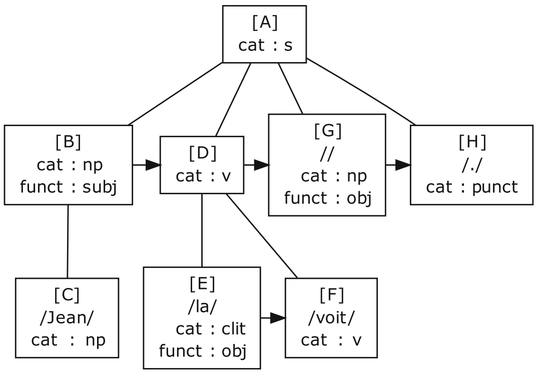


Fig. 1 Syntactic tree for sentence (1)

1.1 A Basic Example

1.1.1 Syntactic Tree

In IGs, the parsing output of a sentence is an ordered tree where nodes represent syntactic constituents described by feature structures.

An example of a syntactic tree for sentence (1) is shown in Fig. 1.[2] Immediate dominance relations are drawn with solid lines and linear order (immediate precedence between sisters) with arrows. We use labels (written with brackets) to refer to nodes of figures in the text, but these labels play no role in the formalism.

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

‘John sees it.’

Each leaf of the tree carries a string called its **phonological form**. This string is empty for nodes used as traces and nonempty for leaves where some word is anchored (phonological form is written between ‘/’ symbols). In our example, [G] carries an empty phonological form and [C] carries the phonological form *“Jean”*. The **phonological form** of a node N which is not a leaf is the concatenation of the phonological forms of its daughters; the phonological form of a tree is the phonological form of its root (*“Jean la voit.”* for the tree of Fig. 1).

1.1.2 Polarized Tree Descriptions

A polarized tree description is a set of constraints both on the tree structure and on the feature structures of syntactic trees. In the following, we will make a distinction[3] between:

– polarized tree description (PTD) which are structures used in the parsing process; a PTD may be not connected;

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

[3] This distinction relies only on the usage and not on the definition: formally, an EPTD is a PTD.

- elementary polarized tree description (EPTD) which are the elementary structures described in the grammar: in the following all EPTDs are connected.

In Fig. 2, there are four EPTDs but we consider the whole set of nodes and relations as one PTD in the parsing process.

The three main kinds of constraints we consider are:

- Constraints on the tree structure of the syntactic tree are expressed with relations. In Fig. 2, [A3] dominates immediately [B3] (solid line): [A3] must be the mother node of [B3] in a syntactic tree which satisfies this constraint. The precedence relation (dashed arrow) between nodes [D2] and [G2] is not necessarily immediate: [D2] must be to the left of [G2] but any number of intermediate nodes between [D2] and [G2] are allowed in the final syntactic tree. This last relation is an example of underspecified relation.
- Constraints on features of the syntactic tree are expressed by polarized features in PTD. These features can be underspecified: in node [F2], `cat` $\sim$ `aux`$|$`v` means that the corresponding node in a syntactic tree must have the feature `cat : aux` or the feature `cat : v`; in node [B1], `funct <- ?` constraints to have the feature `funct` without giving constraints on the corresponding value.
- Saturation constraints are expressed with polarities. In a PTD each feature carries a polarity. These polarities are used to control the interaction of nodes with their environment. In the polarity system we use, a positive (written $\rightarrow$) polarity describes an available resource; it must be associated with a dual negative (written $\leftarrow$) one which describes an expected resource. In Fig. 2, when building a syntactic tree, the positive feature `cat -> s` of node [A3] is associated with the negative feature `cat <- s` of node [A4]. A virtual polarity (written $\sim$) expresses that some context is required: it must be associated to another (non-virtual) feature to be saturated. In Fig. 2, the virtual feature `cat` $\sim$ `v` in node [D2] is associated with the `cat` feature of node [D3] when the model of Fig. 1 is built.

Some other constraints can be used in a PTD: for instance, the node [G2] is declared as an **empty node** (with a dashed border): this requires that the phonological form of the corresponding node in a syntactic tree is empty.

1.1.3 Syntactic Tree as Saturated and Minimal Models of a PTD

Formally, the link between PTDs and syntactic trees uses the notion of model. We say that the syntactic tree of Fig. 1 is a model of the PTD of Fig. 2 because there is a function (called interpretation function) from nodes of the PTD to nodes of the syntactic tree which respects saturation and minimality constraints (these constraints are formally defined in next section). In our example the interpretation function is:

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

We can observe that components of PTD are superposed in the syntactic tree: nodes [A2], [D2], [G2] and [F2] are merged, respectively with nodes [A3], [D3], [G3] and [F3] to produce nodes [A], [D], [G] and [F] in the syntactic tree.

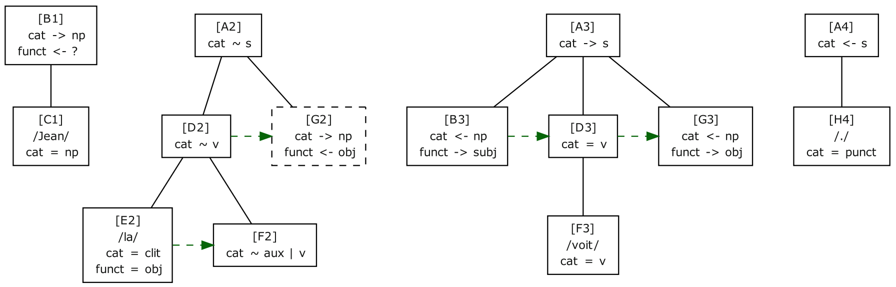


Fig. 2 PTD associated with the words of sentence *"Jean la voit."*

1.2 Node Merging

Whereas the notion of model gives a direct link from PTDs to syntactic trees, it can be useful to consider a more operational view of model building. In order to do this, we define an atomic operation on PTDs named **node merging**. Let $\mathcal{D}$ be a tree description and N and M two distinct nodes of $\mathcal{D}$. The tree description $\mathcal{D}_{\{N=M\}}$ is the description $\mathcal{D}$ where the two nodes N and M are replaced by a unique node NM and the relations are modified accordingly; the polarized features of node NM is obtained from polarized features of both nodes N and M by unification of feature values and by composition of polarities.

With this definition, we can verify the two important properties:

– every syntactic tree which is a model of $\mathcal{D}_{\{N=M\}}$ is a model of $\mathcal{D}$;
– if $\mathcal{D}$ is unsaturated and $\mathcal{T}$ is a model of $\mathcal{D}$ then there are two nodes N and M such that $\mathcal{T}$ is a model of $\mathcal{D}_{\{N=M\}}$.

The first property expresses that node merging is a safe way to construct models.

The second one expresses that every model $\mathcal{T}$ of a tree description $\mathcal{D}$ can be constructed by: first, application of several node mergings on D to build step by step a saturated PTD $\mathcal{D}'$, then verification that $\mathcal{T}$ is a model of D'.

In practice, when two nodes are merged, some propagation rules can be applied to simplify the PTD obtained. For instance, if N has a mother node N' and M has a mother node M' then in the description $\mathcal{D}_{\{N=M\}}$, N' and M' are two mother nodes of the same node NM. Hence, a syntactic tree which is a model of $\mathcal{D}_{\{N=M\}}$ is necessarily a model of $\mathcal{D}_{\{N=M\}\{N'=M'\}}$ and so we can safely apply node merging on nodes M' and N'.

For instance, in Fig. 2, if nodes [F2] and [F3] are merged, then [D2] and [D3] have to be merged also (as mother nodes of [F2] and [F3]); but then [A2] and [A3] have to be merged also (as mother nodes of [D2] and [D3]).

1.3 Polarized Features to Control Syntactic Composition

The notion of polarity represents the core of the IG formalism. In the next section, we give the general definition of a polarity system and we give here only the intuitive meaning of the system used in our implemented grammars.

1.3.1 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 feature when the model is built.

This mechanism is intensively used. It is used similarly as in CGs, 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 IGs to deal with other kinds of interactions. For instance:

- to handle pairs of grammatical words like *ne/pas*, …(see below Sect. 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 full verbs.

1.3.2 Virtual Polarities

Recently, a new polarity was added which is called **virtual** (written $\sim$). To be saturated, 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 Sect. 4.3 for an example;
- to express context constraints on nodes around the positive and negative parts of a description; it allows for a control on the superposition mechanism: in Fig. 2, the three nodes [A2], [D2] and [F2] with virtual `cat` polarities describe the context in which the clitic *"la"* must be used; these three nodes require that three other non-virtual nodes compatible with [A2], [D2] and [F2] exist somewhere else in the PTD; 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).

1.3.3 Polarities at the Feature Level

A difference with respect to other formalisms using polarities is that, in IGs, 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).

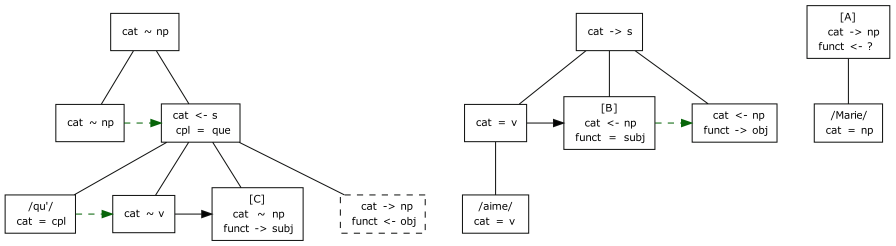


Fig. 3 PTD for the sequence of words *"qu'aime Marie"* before superposition

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

Subject inversion is one of the typical usage of such interactions, that implies more than two nodes. 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. IG allows different solutions. One of them is to use two independent interactions: between the subject and the verb on one hand; and on the other hand between the subject and some other word which is responsible for subject inversion.

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

'John whom 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 Fig. 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.

1.4 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 in nodes R and L. They cannot say anything about the environment of both nodes.

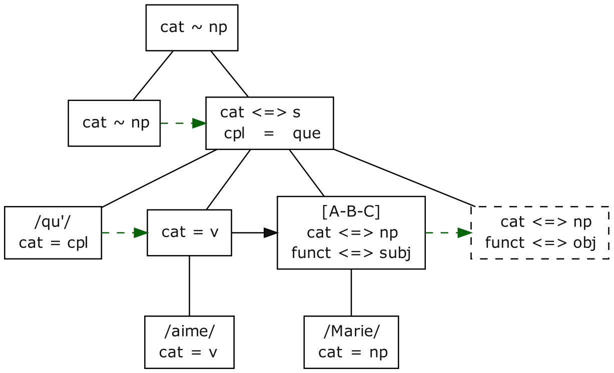


Fig. 4 PTD for the sequence of words *"qu'aime Marie"* after superposition

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.

In IGs, the syntactic composition is much more flexible: we can merge any two nodes of a PTD. Then, the propagation of the constraints 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.

Let us consider the sentence (5).

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

'John knows the author of it.'

The clitic pronoun *"en"* provides the object *"auteur"* of the verb *"connaît"* with a noun complement. Our French lexicon gives the EPTD of Fig. 5 to represent the syntax of this usage of the clitic pronoun *"en"*. In this EPTD, [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, [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 Figs. 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 constraints and by applying some other node mergings to saturate virtual polarity of EPTD of Fig. 5, we obtain the partial superposition of the two PTDs (Fig. 7). Note that there are nine atomic operations of node merging during this composition.

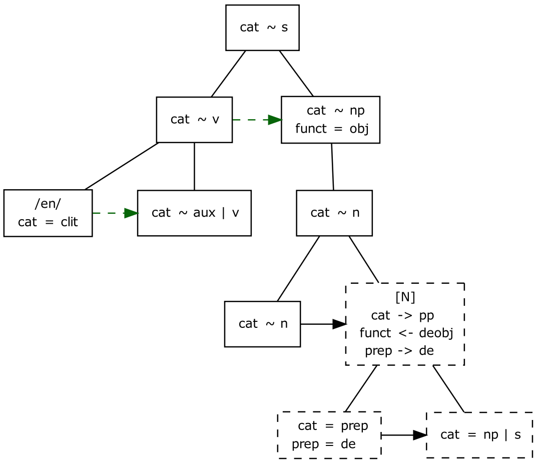


Fig. 5 EPTD representing the syntax of the clitic *"en"*

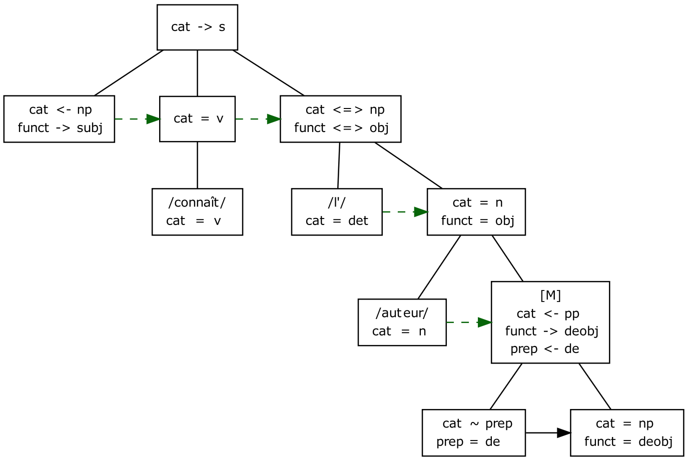


Fig. 6 PTD representing the syntax of the phrase *"connaît l'auteur"*

1.5 Underspecified Structures

With IGs, both dominance and precedence relations can be underspecified: a PTD 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

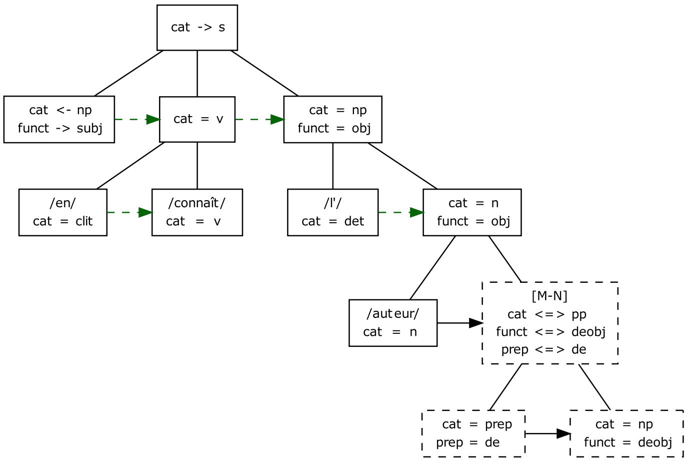
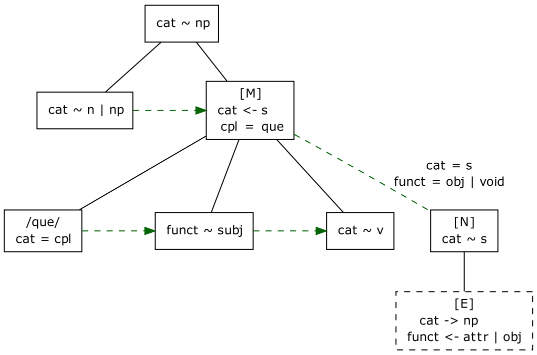


Fig. 7 PTD representing the syntax of the phrase *"en connaît l'auteur"*

Fig. 8 EPTD for the relative pronoun *que*



introduce an unbounded dependency between its antecedent and a verb which has this antecedent as object or adjectival complement: sentences (6) and (7).[4]

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

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

Figure 8 provides a EPTD 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 sentence (6) in

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

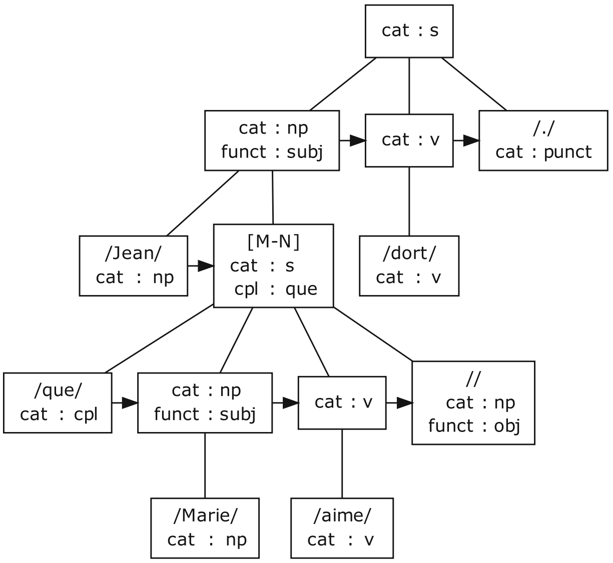


Fig. 9 Syntactic tree for sentence (6)

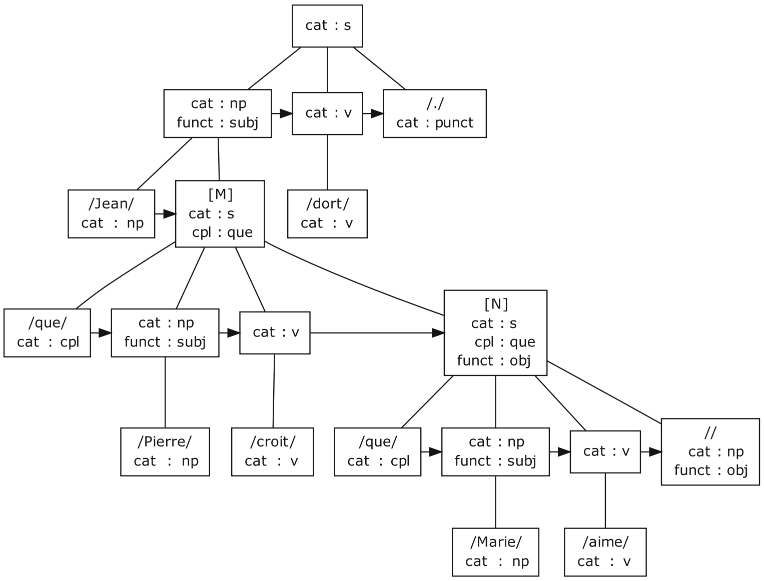


Fig. 10 Syntactic tree for sentence (7)

which the relation is realized by merging [M] and [N], whereas Fig. 10 shows a model for sentence (7) in which the relation is realized by an immediate dominance relation.

In order to deal with island constraints, dominance needs to be controlled. In IGs, 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 dominance

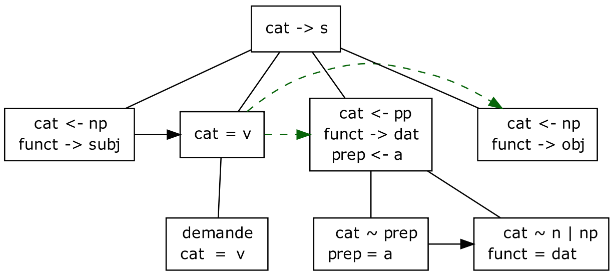


Fig. 11 EPTD for the verb *"demande"*

$M >^* N$ 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 Fig. 8, such a filter (written near to the dashed line of the dominance relation from node [M] to [N]) is used to avoid extraction through nodes that are not of category s.

With underspecification on precedence relation, it is possible to describe a free ordering of some arguments. For instance, both sentences (8) and (9) can be parsed using the same EPTD (Fig. 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 IGs. We define in turn:

- **polarity system**: the set of polarities used in tree descriptions;
- **syntactic trees**: the output syntactic structures of the parsing process;
- **polarized tree descriptions**: the syntactic structures that are used both as inputs at the beginning of the parsing process (EPTDs) and during the parsing process (PTDs).

2.1 Polarity System

2.1.1 The General Case

A **polarity system** is defined as:

- a finite set $\mathcal{P}$ of polarities which contains an element $\perp$ expressing failure;
- an associative and commutative binary operation $\oplus$ on P such that $\perp$ is an absorbing element;
- a subset $\mathcal{N} \subset \mathcal{P}$ such that $\perp \notin \mathcal{N}$.

Polarities that are in $\mathcal{N}$ are called **saturated**, polarities that are not in $\mathcal{N}$ and different from $\perp$ are called **unsaturated**.

The fact that $\oplus$ is an associative and commutative binary operation ensures that the operation can be safely generalized to an operation $\bigoplus$ on nonempty multisets of polarities by iteration of the binary operation:

- $\bigoplus(\{p\}) = p$ for singletons
- $\bigoplus(\mathcal{M} \uplus p) = \bigoplus(\mathcal{M}) \oplus p$

A multiset $\mathcal{M}$ of polarities is said to be **balanced** if $\bigoplus(\mathcal{M}) \in \mathcal{N}$.

The associativity and commutativity of the polarity composition ensures also that models can be constructed with the node merging operation without taking care of the order in which nodes are merged. This claim does not contradict Sygal and Wintner (2009), which shows that “no non-trivial polarity system exists for which grammar combination is associative”. The notion of *grammar combination* is more constrained than our operation of *node merging*: in a grammar combination, two trees are superposed to build a new tree, whereas in a node merging, two nodes of a PTD are merged to build a new PTD without any constraint on the shape of the initial and the final PTDs: we do not impose output structures of node merging to be a tree nor to be connected. Our node merging operation corresponds to the *forest combination* operation described in Sygal and Wintner (2009).

2.1.2 The Polarity System Used in Our Grammars

We describe in this section the polarity system which is used in the implemented French and English grammars. Parsing algorithms of Sect. 6 are also based on this polarity system.

The polarity system used in our grammars is intended to take into account the resource sensitivity of natural languages. $\mathcal{P}$ is $\{=, \rightarrow, \leftarrow, \sim, \leftrightarrow, \perp\}$ and saturated elements are $\{=, \leftrightarrow\}$:

- **neutral** (written $=$): a feature with a neutral polarity is not concerned by the linear resource management: it can be used to saturate a virtual polarity and it is completely neutral with respect to other polarities;
- **positive** (written $\rightarrow$): a feature with a positive polarity describes an available resource;
- **negative** (written $\leftarrow$): a feature with a negative polarity describes a needed resource;
- **virtual** (written $\sim$): a feature with a virtual polarity is unsaturated: it is waiting for an interaction with another non-virtual one; virtual polarities are used for expressing constraints on the context in which a PTD can be inserted;
- **neutralized** (written $\leftrightarrow$): a feature with a neutralized polarity is saturated but it can not be composed with other positive, negative or neutralized polarity;
- **failure** (written $\perp$): this polarity is used when the polarity composition fails.

The polarity composition is defined by the table below:

$\oplus$	$=$	$\rightarrow$	$\leftarrow$	$\sim$	$\leftrightarrow$	$\perp$
$=$	$=$	$\rightarrow$	$\leftarrow$	$=$	$\leftrightarrow$	$\perp$
$\rightarrow$	$\rightarrow$	$\perp$	$\leftrightarrow$	$\rightarrow$	$\perp$	$\perp$
$\leftarrow$	$\leftarrow$	$\leftrightarrow$	$\perp$	$\leftarrow$	$\perp$	$\perp$
$\sim$	$=$	$\rightarrow$	$\leftarrow$	$\sim$	$\leftrightarrow$	$\perp$
$\leftrightarrow$	$\leftrightarrow$	$\perp$	$\perp$	$\leftrightarrow$	$\perp$	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$

With the previous definition it is easy to verify that a multiset $\mathcal{M}$ of polarities is balanced if and only if:

– $\bigoplus(\mathcal{M})$ is $=$ which means that it contains no positive, no negative and a least one neutral;
– or $\bigoplus(\mathcal{M})$ is $\leftrightarrow$ which means that it contains either exactly one positive and one negative polarity or only one neutralized polarity.

In the table below, we describe the different ways to obtain a balanced multiset:

$\#_{=}$	$\#_{\sim}$	$\#_{\rightarrow}$	$\#_{\leftarrow}$	$\#_{\leftrightarrow}$	$\bigoplus$
$n \geq 1$	$n \geq 0$	0	0	0	$=$
$n \geq 0$	$n \geq 0$	1	1	0	$\leftrightarrow$
$n \geq 0$	$n \geq 0$	0	0	1	$\leftrightarrow$

2.1.3 Comparison With Kahane's Polarities

Kahane (2006) proposes Polarized Unification Grammars (PUG), a grammatical framework, where syntactic composition is viewed as superposition of graphs under the control of polarities. PUG can be instantiated with different polarity systems. In Kahane (2006) , the polarity system given as an example is $\mathcal{P}' = \{\boxtimes$[5]$, +, -, \square, \blacksquare, \perp\}$. With the correspondence given in the table below, we have the same set of saturated polarities ($\mathcal{N}$ is $\{=, \leftrightarrow\}$ and $\mathcal{N}'$ is $\{\boxtimes, \blacksquare\}$)

$\boxtimes$	$+$	$-$	$\square$	$\blacksquare$	$\perp$
$=$	$\rightarrow$	$\leftarrow$	$\sim$	$\leftrightarrow$	$\perp$

This polarity system slightly differs from our system. The difference relies in the composition operation:

$$= \oplus \sim \text{ is } = \qquad \text{whereas} \qquad \boxtimes \oplus \square \text{ is } \square$$

The consequence on the corresponding operation on multiset is given by the comparison of the two tables below:

[5] We use the notation $\boxtimes$ for the grey polarity instead of a grey box of the original paper.

$\#_{=}$	$\#_{\sim}$	$\#_{\rightarrow}$	$\#_{\leftarrow}$	$\#_{\leftrightarrow}$	$\bigoplus$
$n \geq 1$	$n \geq 0$	0	0	0	$=$

$\#_{\boxtimes}$	$\#_{\square}$	$\#_{+}$	$\#_{-}$	$\#_{\blacksquare}$	$\bigoplus$
$n \geq 1$	0	0	0	0	$\boxtimes$

Due to this difference, there is no automatic translation of an IG with a polarity system $\mathcal{P}$ to one (or from one) with the polarity system $\mathcal{P}'$. But the actual grammars can be reformulated with Kahane's system by replacing some $=$ polarities (that are used to saturate structures with $\sim$ polarities) by $\leftrightarrow$ polarities.

2.2 Syntactic Trees

2.2.1 Feature Signature

A **feature signature** is defined by:

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

2.2.2 Feature and Feature Structure

Features are built relatively to a feature signature. A **feature** is a pair (f, v) where $f \in \mathcal{F}$ and $v \in \mathcal{A}_f$; the feature (f, v) is written $f\ :\ v$ in the rest of the paper and in the figures. A **feature structure** is a set of features with different feature names.

2.2.3 Syntactic Tree

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

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

In syntactic trees, immediate dominance is written $M \gg N$ (this means that M is the mother node of N), immediate precedence between sisters is written $M \lll N$ (this means that M and N have the same mother and that M is just before N in the sisters ordering).[6]

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':

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

Let $\lll^+$ denote the transitive closure of the relation $\lll$.

6 We use double symbols to avoid confusion with relations that are defined later for PTDs.

We extend the definition of phonological form to all nodes: for a node M which is not a leaf, we define the **phonological form** (written $PF(M)$) to be the string built by concatenation (with an extra white space between the two strings if they are both nonempty) of the phonological form of its daughters. The phonological form of a syntactic tree is the phonological form 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 daughters with empty phonological form, 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.

2.3 Polarized Tree Description

2.3.1 Polarized Feature and Polarized Feature Structure

Whereas features in syntactic trees are defined by a pair name-value, in a tree description, feature values can be underspecified (with a disjunction of atomic values) and a polarity is attached to each feature. The features are build on a feature signature on which we add polarity systems: for each feature name $f \in \mathcal{F}$, in addition to the set of $\mathcal{A}_f$ of atomic values, we consider a polarity system $\mathcal{P}_f$.

Formally, a **polarized feature** is defined by a triple of:

- a feature name f taken from $\mathcal{F}$,
- a polarity taken from $\mathcal{P}_f$,
- a feature value which is a disjunction of atomic values taken from $\mathcal{A}_f$; 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 from $\mathcal{A}_f$.

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

It is also possible to give additional constraints on feature values with co-references. A co-reference is an integer called an **index** (written $\langle i \rangle$); for instance `mood =` $\langle 2 \rangle$ `ind|subj` is a co-referenced feature. The scope of these co-references is the PTD: a feature value can be shared between two different nodes of the same PTD and it is defined up to renaming of indices. When we consider the disjoint union of a multiset of EPTDs, we suppose that indices are local to each EPTD of the multiset. Hence, in practice, some renaming may be necessary when disjoint union are built.

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

2.3.2 Filtering Feature and Filtering Feature Structure

Filtering features are used to represent constraints on dominances. A **filtering feature** is a polarized feature with a neutral polarity and without co-reference. A **filtering feature structure** is a set of filtering features with different feature names.

A feature structure φ is said to be **compatible** with a filtering feature structure Ψ if, for each feature name f defined in Ψ with value v, φ contains the feature f with a value included in the disjunction v.

So far, we have defined three different kinds of features. We give in the table below a global view of the differences:

	Feature	Polarized feature	Filtering feature
Localization	nodes in syntactic trees	nodes in PTDs	dominance relations in PTDs
Polarity	no	any	=
Value	atomic	disjunction	disjunction
Co-references	no	yes	no

2.3.3 Node Description

A **node description** is:

– a polarized feature structure;
– an optional node type.

Node types express constraints on the phonological form of nodes in the model. Each node can have one of these three types:

– **anchor**(w) (where w is a nonempty string): this type will constraint the image of the node by the interpretation function to be a leaf with the phonological form w;
– **empty**: this type constrains the image to have an empty phonological form (empty nodes are drawn with dashed border in figures);
– **nonempty**: this type constrains the image to have a nonempty phonological form.

2.3.4 Polarized Tree Description

A **polarized tree description** (PTD) is defined by a set of node descriptions and a set of relations on these nodes. We consider four types of relation between nodes in PTDs:

Immediate 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 the image of N is the leftmost (resp. rightmost) daughter of image 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 M).

Dominance
$M >^* N$ constrains the image of N to be in the subtree rooted at the image of M.[7] A dominance can also carry an additional constraint on 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.

Immediate 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.

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.

The **disjoint union** $\coprod \mathcal{M}$ of a multiset $\mathcal{M}$ of PTDs is defined as the disjoint union of the set of nodes and relations (with co-references renaming when necessary).

3 Syntactic Trees as Models of PTDs

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

3.1 Syntactic Trees as Models of PTDs

A syntactic tree $\mathcal{T}$ is a model of a PTD $\mathcal{D}$ if there is an **interpretation function** $\mathcal{I}$ from nodes $\mathcal{ND}$ of $\mathcal{D}$ to nodes $\mathcal{NT}$ of the syntactic tree $\mathcal{T}$ such that:

Immediate dominance adequacy

- if $M, N \in \mathcal{ND}$ and $M > N$ then $\mathcal{I}(M) \gg \mathcal{I}(N)$;
- if $M, N \in \mathcal{ND}$ and $M > \bullet N$ then $\mathcal{I}(M) \gg \mathcal{I}(N)$ and N is the leftmost daughter of M;
- if $M, N \in \mathcal{ND}$ and $M > N\bullet$ then $\mathcal{I}(M) \gg \mathcal{I}(N)$ and N is the rightmost daughter of M;
- if $M, N_1, \ldots, N_k \in \mathcal{ND}$ and $M > \{N_1, \ldots, N_k\}$ then $\mathcal{I}(M)$ has exactly k different daughters $\{\mathcal{I}(N_1), \ldots, \mathcal{I}(N_k)\}$

Dominance adequacy

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

[7] 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.

- if $M, N \in \mathcal{ND}$ and $M >_{\Psi}^{*} N$ then $\mathcal{I}(M) \gg^{*} \mathcal{I}(N)$ and for each node P in $path(\mathcal{I}(M), \mathcal{I}(N))$, the feature structure φ of P is compatible with Ψ.

Immediate precedence adequacy

- if $M, N \in \mathcal{ND}$ and $M \prec N$ then $\mathcal{I}(M) \lll \mathcal{I}(N)$. Note that by definition of the relation $\lll$ in syntactic trees, this impose that $\mathcal{I}(M)$ and $\mathcal{I}(N)$ have a common mother node.

Precedence adequacy

- if $M, N \in \mathcal{ND}$ and $M \prec^{+} N$ then $\mathcal{I}(M) \lll^{+} \mathcal{I}(N)$. This also impose that $\mathcal{I}(M)$ and $\mathcal{I}(N)$ have a common mother node.

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 the feature f exists in N and the disjunction associated with f contains v or N does not contain the feature name f;
- if $M, N \in \mathcal{ND}$ are two nodes which 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 $M \in \mathcal{ND}$ has type anchor(w) then $\mathcal{I}(M)$ is a leaf and $PF(\mathcal{I}(M)) = w$;
- if $M \in \mathcal{ND}$ has type empty then $PF(\mathcal{I}(M)) = \epsilon$;
- if $M \in \mathcal{ND}$ has type nonempty then $PF(\mathcal{I}(M)) \neq \epsilon$.

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 balanced.

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 $M \in \mathcal{ND}$ is a leaf node with a nonempty phonological form *phon*, then $\mathcal{I}^{-1}(M)$ contains exactly one node of type anchor(w).

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, immediate dominance relation creation, feature creation, and phonological form creation.

3.2 Polarized Grammars

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

We say that a syntactic tree $\mathcal{T}$ is a **parse tree** of a sentence S if:

- $\mathcal{T}$ is a model of $\coprod \mathcal{M}$ for some multiset $\mathcal{M}$ of EPTDs,
- $PF(\mathcal{T}) = S$.

An interaction grammar $\mathcal{G}$ is said to be **lexicalized** if each EPTD contains at least one node of type anchor(w).

An interaction grammar $\mathcal{G}$ is said to be **strictly lexicalized** if each EPTD 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 EPTDs 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 IGs 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 IGs 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.’

(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	company		decides	alone.

‘The director in no company decides alone.’

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

‘John is at the head of no company.’

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

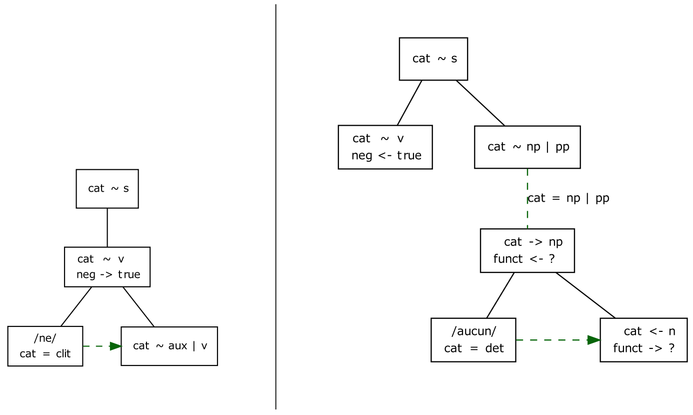


Fig. 12 EPTDs associated with the particle *"ne"* and the determiner *"aucun"*

A classical way of modelling the pairing of words is to anchor a common syntactic structure with both words, but it presents the drawback of making us leave strongly lexicalized grammars: from a computational point of view, it is more convenient to manipulate EPTDs anchored by exactly one word.

IGs allow us to model the pairing of words in the framework of strongly lexicalized grammars by using special polarized features. In the example of *"ne . . . aucun"*, the linking of the two words is realized by a polarized feature `neg`. The EPTDs from Fig. 12, associated with the words *"ne"* and *"aucun"*, show how the polarized feature `neg` acts. The word *"ne"* puts 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 EPTD, there is a constraint in the underspecified dominance relation that forbids the sentence (15) from being accepted. The EPTDs from Fig. 12 allow all sentences above to be correctly parsed.

4.2 Constrained Dominance Relations Modelling Long-Distance Dependencies

Underspecified dominance relations are used to represent unbounded dependencies and the filtering 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'*
	John	in	the	company	of	whom	Mary	knows	that	the

ingénieur	*travaille*	□	*est*	*malade.*
engineer	works		is	sick.

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

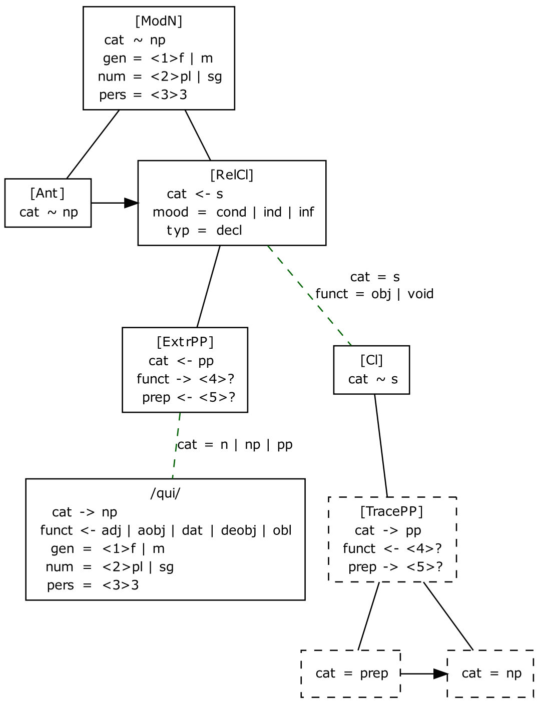


Fig. 13 EPTD associated with the relative pronoun *"qui"* used in an oblique complement

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

(18)	**Jean*	*[dans*	*l'*	*entreprise*	*qui*	*appartient*	*à*	**qui***]*	*Marie*
	John	in	the	company	which	belongs	to	whom	Mary
	travaille	□	*est*	*malade.*					
	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 a dominance relation in the EPTD describing the syntactic behaviour of the relative pronoun *"qui"* on Fig. 13. The dominance relation links the node [RelCl] representing the relative clause *"[dans l'entreprise de* **qui***] Marie sait que l'ingnieur travaille* □*"* and the node [Cl] representing the clause *"que l'ingnieur 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 Fig. 13 with a 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. As a consequence, the sentence (18) is rejected.

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 IGs Perrier (2005). Modifier adjunction was performed in the same way as in several formalisms (CGs, 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 Nasr (1995), we have introduced the virtual polarity $\sim$. This polarity allows for *sister adjunction*, that is 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 EPTD 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 EPTD.

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 the community of linguists (Carston and Blakemore 2003; Godard 2005). 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. Here are sentences that 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*
The	destruction	of	the	bus station	by		bombs	and	of	the
gare ferroviaire	*par*	*les*	*tanks]*	*rend*	*l’*	*accès*	*à*	*la*	*ville*	*difficile.*
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*	*sœur*	*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 (24), 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 gapping. Here, the gap, which is represented by the □ symbol, corresponds to the elided verb *“aime”*.

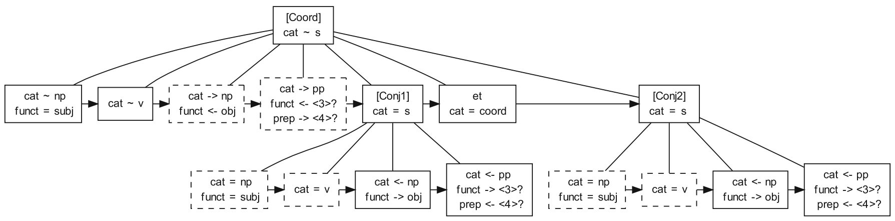


Fig. 14 EPTD associated with the conjunction *"et"* used for coordinating argument clusters

Le Roux and Perrier (2006) propose to model coordination in IGs with the notion of polarity (Le Roux 2007). We propose to model coordination in IGs with the notion of polarity The interface of a PTD is defined as the nodes that carry non saturated polarities and their structural relations. 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, ranked the merge of two interfaces cannot be modelled directly in IGs 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.

Let us illustrate these principles with the conjunction *"et"* used for coordinating argument clusters such as in sentence (25). In this sentence, the argument clusters that have to be coordinated are *"des fleurs à Marie"* and *"des bonbons à Pierre"*. If we represent their syntactic structure with IGs, we obtain two PTDs with the same interface constituted of two ordered nodes:

– a first node representing a noun phrase with two polarized features: `cat -> np` and `funct <- ?`,
– a second node representing a prepositional phrase with three polarized features: `cat -> pp`, `funct <- ?` and `prep -> a`.

Because of the same interface, the two argument clusters can be coordinated by the conjunction *"et"*. The appropriate EPTD is given by Fig. 14. The lower parts of the EPTD are two identical subtrees rooted at nodes [Conj1] and [Conj2]. They are used to saturate the interfaces of the two conjuncts. Then, we obtain a EPTD representing the coordinated expression *"des fleurs à Marie et des bonbons à Pierre"*. The interface of this EPTD is the higher part of the EPTD associated with the conjunction: the node [Coord] with its four leftmost daughters. Finally, this interface is saturated in the interaction with the phrase *"Jean donne"*.

Of course, we need a different EPTD for every type of interface, which entails an important number of entries for *"et"* in the lexicalized grammar. The same problem occurs in CGs where the conjunction has the generic type $X\backslash X/X$ with X being instantiated with a concrete type taken from a list of possible types. The type X corresponds to the interface of conjuncts used with IGs.

The inflation of entries for the coordination conjunctions is a drawback for parsing but this can be limited because it is possible to foresee the conjuncts of a coordinated expression and their common interface. Then, it is easy to select the appropriate EPTD for the conjunction.

5 Comparison With Other Formalisms

Currently, there exists no linguistic formalism 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 generalizations as much as possible while remaining tractable. Remaining tractable means being able to build large scale grammars and efficient parsers. From this perspective, the number of relevant formalisms is not very important: among the most well-known and largely used, there are LFG, HPSG, TAG or CCG. The comparison of IGs with other formalisms will highlight some of its strong features.

5.1 Categorial Grammars

The list of linguistic formalisms above mentions CCG (Combinatory Categorial Grammars) (Steedman 2000). CCG is currently the most famous instance of the CG family. 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 (Steedman 1985), and it is used for parsing large corpora with efficient polynomial algorithms and large scale grammars (Hockenmaier 2003; Clark and Curran 2004). Now, we aim at comparing IGs with CGs in a general way but, when it will be necessary to be more precise, we will choose CCG as the most prominent instance of CGs.

IGs share with CGs 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. Referring again to the distinction between two approaches for syntax introduced by Pullum and Scholz (2001), we observe that CGs use a generative-enumerative syntactic (GES) framework whereas IGs use a model-theoretic syntactic (MTS) framework. In other words, CGs derive all acceptable sentences of a language from a finite set of axioms, the lexicon, using a finite set of deduction rules. IGs associate a sentence with a set of constraints, which are solved to produce its syntactic structure.

In CGs, the syntax of a natural language expression is represented by a derivation tree. The type, conclusion of the tree derivation, represents the interface of the expression in the sense given in the Sect. 4.4: it summarizes its ability to interact with other expressions in the form of a logical type.

Perrier (2001) proposes a method for transposing grammars from the GES to the MTS framework under some conditions. This method applies to CGs and can be used to compare IGs with CGs by putting them in the same MTS framework. The precise description of such a translation goes beyond the goal of this article; we only present the main conclusions.

- Each syntactic type of CGs corresponds to an EPTD with a very constrained shape. Polarities are attached to nodes, which are either positive or negative. Immediate dominance relations always go from positive nodes to negative nodes. For dominance relations, that is the contrary.
- Unlike general IGs, word order is closely linked to dominance relation. Left CG implication expresses that a phrase expects a constituent on its immediate left. There is a symmetric rule for right implication.
- Syntactic composition is guided by the system of polarities but it uses superposition of PTDs in a very limited way. Node merging is restricted to pairs of dual nodes. It is not possible to express passive constraints on the environment of a syntactic object, as we do in IGs using nodes with virtual and neutral features.

These features give more rigidity to CGs than general IGs. It is especially obvious for the kernel of every CG deductive system, which is constituted of the backward and forward application rules. These rules allow constituents only to be composed by concatenation, which is not sufficient to express the complexity of natural languages.

Usually, flexibility is introduced by adding specific inference rules to the two basic rules. In CCG, there are forward and backward rules for type raising, composition and crossing composition. Such rules introduce limited associativity and commutativity in the combination of constituents. The counterpart is that they are a source of over-generation for CCG.

To limit over-generation, Baldridge and Kruijff (2003) have introduced modalities to control the applicability of combination rules. These modalities are introduced in the lexicon, so that the syntactic behaviour of a word can be more or less constrained. The problem is that one cannot express 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.

IGs do not present such a drawback, because *“yesterday”* is taken as a sentence modifier and it is modelled with virtual polarities according to the method presented in Sect. 4.3.

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

5.2 Unification Grammars

If the main originality of IGs, the concept of polarity, brings it closer to CGs, the view of the syntactic composition as a tree superposition, brings it nearer to Unification

Grammars (UG). The family of UG includes all formalisms for which the mechanism of unification between feature structures occupies a central position.

Head-driven Phrase Structure Grammar (HPSG) Sag et al. (2003) exploits this idea at the maximum: 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 graphs, in which edges are labeled with feature names and leaves with atomic feature values. In this way, unification appears as graph superposition. As in IGs, 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 IGs 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 IGs.

– Graphs are more expressive than trees. In this way, some phenomena are easier to model with HPSG than with IGs. For instance, factorization, which is specific to coordination, is directly represented in HPSG (Mouret 2007), whereas it must be simulated in IGs (see Sect. 4.4).
– Underspecification is more restricted in HPSG than in IGs; it reduces to the underspecification associated with unification. All dominance relations are immediate (completely specified), so that unbounded dependencies are represented with another mechanism: the *slash feature*, the propagation of which allows to model unbounded dependencies.
– Word order is not expressed by linear order between graph nodes but with a specific feature PHON.

Lexical Functional Grammar (LFG) (Bresnan 2001) 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 PTDs. 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 IGs: the concept of *functional uncertainty* is similar to the IG notion of dominance, with the same possibility of constraining the dominance path between nodes without determining its length.

Polarized Unification Grammars (PUGs) (Kahane 2006) are a meta-formalism rather than a formalism, because they aim at representing most usual formalisms in a common framework by polarizing their elementary structures. Then, syntactic composition is viewed in an uniform way as superposition of graphs under the control of polarities. Polarization aims at controlling the saturation of the final structures.

On one hand, PUGs are more general than IGs, because syntactic structures are graphs and not trees, and polarization extends to edges. On another hand, they are more restrictive, because they manipulate completely specified graphs and not graph descriptions with some paths left underspecified.

5.3 Grammars Based on Trees and Tree Descriptions

Tree Adjoining Grammar (TAG) (Abeillé and Rambow 2001) is usually ranked in the UG family, even if it is more tree grammar than UG: the use of unification is limited and contrary to most UG, it cannot be used to superpose structures. Structures only combine by adjunction, which greatly limits the expressivity of the formalism.

TAG is generally used in its lexicalized version (LTAG) but there is no flexibility for choosing the words to which grammatical information is anchored. For clitic pronouns, complementizers, different forms of extraction, syntactic constructions are anchored at verbs and not at grammatical words, when they are present, because there is no way for doing differently without relaxing the formalism (Kahane et al. 2000). The bad consequence is a multiplication of entries for verbs in TAG lexicons.

The rigidity of TAG also appears in modelling constructions where grammatical words have a double action in two different places, which can be distant from each other. This is the case for relative or interrogative words but also for clitic pronouns in French. It is especially obvious for the pronoun *"en"* used a noun complement, as in sentence (5), which is repeated below:

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

'John knows the author of it.'

In this sentence, *"en"* modifies the verb *"connaît"* as a clitic and, at the same time, brings a noun complement to *"auteur"*. TAG are unable to represent this double action; their extension to Multi-Component TAG (MCTAG) (Weir 1988) is an answer to this problem.

Finally, the use of auxiliary trees for modelling modifiers is problematic. It introduces a new level in the syntactic tree, which often appears as artificial and brings spurious ambiguity. Moreover, it forbids the insertion of a modifier inside the constituent that is modified.

(32)	*Jean,*	*cet*	*après-midi,*	*rencontre*	*Marie.*
	Jean,	this	afternoon,	is meeting	Marie.

'John is meeting Marie this afternoon.'

In sentence (32), *"cet après-midi"* modifies the sentence while being put in the middle of it. It cannot be modelled with TAG, where modifiers are put only on the left or on the right of the modified constituent.

In IGs, virtual polarities allow a modifier to be added as a new daughter of the node that it modifies without changing the rest of the syntactic tree, in which the modified node is situated. This operation, called *sister adjunction*, is used in some formalisms like Dependency Grammars (DGs). This way of modelling modifiers is very flexible: example (32) is treated without difficulty but that is also the case for more complicated phenomena, such as parenthetical clauses.

A way of relaxing the rigidity of TAG consists in changing the perspective to putting the concept of tree description in the center of the formalism. Rambow et al. (2001) revisit TAG from this perspective and propose to relax the formalism in order to extend its expressivity. They present a new formalism, Description-tree Substitution Grammars (DSGs), in which the basic objects are tree descriptions. Tree descriptions are composed with an operation called *generalized substitution* which generalizes adjunction but which is not as general as tree superposition.The main restriction is that two immediate dominance relations coming from two different tree descriptions are not allowed to merge.

The major difference with respect to IGs lies in the form of constraints put on syntactic composition: DSGs use restrictions on the form of superposition, whereas IGs use only polarities to constrain tree superposition.[8] Since polarities are put freely on tree descriptions, IGs offer more flexibility for modelling linguistic phenomena: for instance, they can represent extraction in a manner similar to DSGs, using constraint dominance relations, but they can also represent modifiers with virtual polarities in a manner that does not fit in with DSGs.

5.4 Dependency Grammars

All previous formalisms share with IGs their inclusion in phrase structure grammars. They differ from Dependency Grammars (DGs) (Nivre 2005) in their basic concept: *constituent* for the first ones, and *dependency* for the second ones.

When we compare IGs with DGs, we are confronted again with this fundamental difference and their well-known consequences. Nevertheless, behind the concept of dependency, there is some proximity to IGs. 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 DGs (Tesnière 1959). Even if there is no explicit notion of polarity in DGs, this underlies the notion of dependency. The potentiality of two words to establish a dependency between themselves can be expressed by equipping them with polarities. The type of polarities depends on the nature of the dependency:

– If the dependency is required by the *régissant* and corresponds to its valency, we equip it 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.
– If the *subordonné* is a modifier of the *régissant*, we equip it with a virtual feature, the *régissant* having a non virtual feature with the same value.

In this way, it is possible to translate a dependency grammar into an IG. In the resulting IG, the EPTDs have the very simple shape of trees of depth one. As a consequence, PTD superposition reduces to tree hanging. The difficulty is to model word order

[8] It is possible to simulate the generalized substitution of DSGs inside IGs by decorating primitive tree descriptions in the following way:

– In a parent-child description, the root is equipped with a positive polarity, whereas all substitution nodes are equipped with a negative polarity.
– In a domination description, the top node is equipped with a positive polarity and the bottom node with a neutral polarity.

in presence of cross-dependencies, which entail non projectivity of grammars. For this, we have to relax the IG constraint that every PTD node represents a continuous constituent.

Conversely, from a parsing with an IG, it is possible to extract dependencies between words (Marchand et al. 2009). The operation of node merging that occurs between two PTDs anchored by two different words for realizing certain polarity saturations can be viewed as the achievement of a dependency between the two words. In this way, for every parsing of a sentence with IG, we obtain a dependency graph between its words, which is refinement of the dependency tree usually expected.

6 Parsing With Polarities

Polarities play a central role in the definition of the formalism and in linguistic modeling. But another important aspect of polarities is that they bring original parsing methods which are orthogonal to usual ones.

Polarities can be useful both for lexical disambiguation and for deep parsing.

These methods focus on strictly lexicalized IGs. Nevertheless, it is easy to transform any lexicalized grammar into an equivalent strictly lexicalized grammar with the mechanism used in Sect. 4.1.

With a strictly lexicalized grammar, if a $\mathcal{T}$ is a syntactic tree for a given sentence then each word contributes to the model with exactly one EPTD.

Hence, the parsing process can be divided in two steps: first, select for each word of the sentence one of the EPTDs which contains this word as its anchor; then build a syntactic tree which is a model of the disjoint union of EPTDs chosen in the first step. The choice of one EPTD for each word of the sentence is called a **lexical selection**.

6.1 Complexity

The general parsing problem for IGs 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 IGs. Intuitively, the complexity has two sources:

- **Lexical ambiguity** In a lexicalized IG, each word of the lexicon can be associated to several EPTDs. 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 disjoint union of EPTDs. Building a model is equivalent to finding a partition on the set of nodes of the PTDs such that each equivalent class is interpreted as a model node. Once again, there is an exponential number of possible partitions.

The next two subsections address these two sources of ambiguity. We are looking for algorithms which behave in an interesting way for real NLP grammars. For instance, the IG formalism can be used to define a grammar without any unsaturated 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 recall a method which is formalized in a previous paper (Bonfante et al. 2004) 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 Boullier (2003): we consider an abstraction of our syntactic structures for which parsing is more 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 IGs, we consider as an abstract view of an EPTD the multiset of positive and negative features present in the EPTD. Then, a lexical selection is valid in the abstract level if the union of the multiset associated to EPTDs contains the same number of positive and negative polarities. In other words, we focus on the saturation requirement (defined in 3.1) of the set of constraints given on models, which reduces to neutralization if we forget virtual polarities.

The parsing at this abstract level can be efficiently implemented with acyclic finite state automata (FSA). For each pair (f, v) of a feature name and a feature value, an automaton is build with EPTDs 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 pair (f, v) along every paths from initial state to the current state. Finally, only lexical selections that end in a state labelled with 0 should be kept.

The automaton obtained has $O(n^2)$ nodes for a sentence of lenght n: the set of nodes of this automaton can be split in layers: a layer is the set of nodes that are reachable by prefix of the sentence of length k. Now, let c be the maximum number of positive or negative polarity in ETPDs of a grammar. In the FSA built for a sentence, there are at most $2 \cdot c \cdot k$ nodes in each layer. So, for a total length of n, the FSA contains at most $\Sigma_{k=1}^{n} 2 \cdot c \cdot k$, that is $O(n^2)$.

The fact that feature values can be disjunction of atomic values in EPTDs causes the automata to be nondeterministic. We turn them into deterministic ones using intervals of integers instead of integers in states of the automaton. In this case, number of state is $O(k^2)$ in each layer and so $O(n^3)$ for a sentence of length n.

Such an automaton is built for each possible pair (f, v), then a FSA intersection of the set of automata describes the set of lexical selections that are balanced. The number of automata to intersect is a constant s given by the feature signature. The worst case size for the intersection is then $O(n^{3 \cdot s})$. Hence, the filtering stage can be done in polynomial time.

We can not have any warranty on the performance of the filtering stage in the general case (grammar without any unsaturated polarity being the worst case, because filtering stage does not remove any lexical selection). But 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 sentence (33) the number of lexical selections reduces from 578,340 to 115 (in 0.08 s).

(33)	*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 ongoing works try to apply some finer filters on automaton. In Bonfante et al. (2006), a specialized filter is described dealing with coordination for instance. For each EPTD for a symmetrical coordination, this filter removes the EPTD if it is not possible to find two sequences of EPTD on each side of the coordination with the expected multiset of polarities.

6.3 Deep Parsing

Deep parsing in IGs is a constraint satisfaction problem. Given a list of EPTDs, we have to find the set of models of the disjoint union of EPTDs which respect word order of the input sentence.

6.3.1 The Incremental Algorithm

As already said, there is an exponential number of possible choices of couples of nodes to merge. The current implementation of IGs is based on an incremental algorithm with scans the sentence word by word, trying to mimic the human reading. The algorithm uses a kind of shift/reduce mechanism; we start with an empty PTD and then we used the two rules:

REDUCE: in the current PTD, try the different ways to neutralize a pair of positive/negative features;
SHIFT: add the next EPTD to the current PTD.

In our implementation, the search space of the algorithm is controlled by a bound inspired by psycholinguistics motivations to guide the parsing. This notion of bound is used in a very similar spirit in Morrill’s works (Johnson 1998; Morrill 2000).

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, this informal hypothesis can be expressed easily with the help of polarities: unresolved dependencies are positive and negative polarities for which no dual feature have been found so far.

We bound the number of positive or negative polarities and then the [SHIFT] operation is accepted only if the current PTD contains a number of positive or negative polarities which is lower than a given bound. With this kind of bound, the parsing is polynomial. The drawback is that this parsing is of course not complete. But we can observe that real sentences can be parsed with a low bound and that this algorithm can be use in practice.

6.3.2 Other Algorithms

The well-known CKY parsing algorithm for CFG can be adapted to IGs. 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.

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 main drawback is that the size of that chart can explode. Another drawback is that it is designed to find only models that follow some projectivity condition. 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.

We have also considered an algorithm inspired by the classical Earley parsing algorithm for CFG. The algorithm for IGs is described in Le Roux (2007), Marchand (2006). This algorithm is still an ongoing work and it must be further optimized.

6.4 Implementation

The IG formalism is implemented in a parser named LEOPAR[9]. This parser is a software which is mainly used as a development tool for grammar writers.

The main notable aspects of the software are the following:

– The parser implements the two stages of parsing: filtering and deep parsing.
– It contains a flexible mechanism to combine several kind of linguistics resources: the link between an external lexicon (independent of the IG formalism) and an unanchored grammar is done through interfaces which are a kind of hypertags.
– Unanchored PTDs can be described with an XML format produced by XMG (Duchier et al. 2004) (an external tool which provides a high level language to build large coverage grammars).
– 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 lexical disambiguation stage and then choose couple of nodes to merge: this interactive mode is useful mainly in grammar testing/debugging.

7 Conclusion

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

9 http://leopar.loria.fr.

One of our fundamental ideas is to combine theory and practice. The formalism of IGs 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 Perrier (2007), which is usable with a lexicon independent of the IG formalism. There exists a lexicon with a large coverage available in the format required by the grammar: the Le*fff* (Sagot et al. 2006). 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 IGs; 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; Perrier (2005) presents a first proposal for the extension of IGs to the semantic level but we can envisage other approaches using existing semantic formalisms such as MRS Copestake et al. (2005) or CLLS Egg et al. (2001).

Acknowledgments We would like to thank Carl Pollard for his useful comments on this work.

References

Abeillé, A., & Rambow, O. (Eds.). (2001). *Tree adjoining grammars: Formalisms, linguistic analysis and processing*. Stanford: CSLI.

Adjukiewicz, K. (1935). Die syntaktische konnexität'. *Studia Philosophica, 1*, 1–27.

Baldridge, J., & Kruijff, G. -J. (2003). Multi-modal combinatory categorial grammar. In *10th Conference of the European chapter of the association for computational linguistics (EACL '2003)*. Budapest, Hungary.

Bonfante, G., Guillaume, B., & Perrier, G. (2003). Analyse syntaxique électrostatique'. *Traitement Automatique Des Langues, 44*(3), 93–120.

Bonfante, G., Guillaume, B., & Perrier, G. (2004). Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *20th International conference on computational linguistics, CoLing 2004, Genève, Switzerland* (pp. 303–309).

Bonfante, G., Le Roux, J., & Perrier, G. (2006). Lexical disambiguation with polarities and automata. In *Lecture notes in computer science*, (Vol. 4094. pp. 283–284), Springer.

Boullier, P. (2003). Supertagging: A non-statistical parsing-based approach. In *Proceedings of the 8th international workshop on parsing technologies (IWPT 03). Nancy, France* (pp. 55–65).

Bresnan, J. (2001). *Lexical-functional syntax*. Oxford: Blackwell.

Carpenter, B. (1998). *Type-logical semantics*. Cambridge, MA: MIT Press.

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

Clark, S., & Curran, J. (2004). Parsing the WSJ using CCG and log-linear models. In *42nd Annual meeting of the association for computational linguistics (ACL '04). Barcelona, Spain* (pp. 103–110).

Copestake, A., Flickinger, D., Pollard, K., & Sag, I. (2005). Minimal recursion semantics—an introduction. *Research on Language and Computation, 3*, 281–332.

Duchier, D., Le Roux, J., & Parmentier, Y. (2004). The metagrammar compiler: An NLP application with a multi-paradigm architecture. In *Second international Mozart/Oz conference, MOZ 2004, Charleroi, Belgium* (pp. 175–187).

Duchier, D., & Thater, S. (1999). Parsing with tree descriptions: A constraint based approach. In *Natural language understanding and logic programming NLULP'99,Dec 1999, Las Cruces, New Mexico*.

Egg, M., Koller, A., & Niehren, J. (2001). The constraint language for lambda structures. *JOLLI, 10*, 457–485.

Godard, D. (2005). Problmes syntaxiques de la coordination et propositions rcentes dans les grammaires syntagmatiques. *Langages, 160*, 3–24.

Hockenmaier, J. (2003). Parsing with generative models of predicate-argument structure. In *41st Annual meeting of the association for computational linguistics (ACL '03). Sapporo, Japan* (pp. 359–366).

Jespersen, O. (1937). *Analytic syntax*. London: Allen and Unwin.

Johnson, M. (1998). Proof nets and the complexity of processing center-embedded constructions. *JOLLI, 7*(4), 433–447.

Kahane, S. (2006). Polarized unification grammars. In *21st International conference on computational linguistics and 44th annual meeting of the association for computational linguistics. Sydney, Australia* (pp. 137–144).

Kahane, S., Candito, M. -H., & de Kercadio, Y. (2000). An alternative description of extractions in TAG. In *Workshop TAG+5, Paris* (pp. 115–122).

Kallmeyer, L. (1999). *Tree description grammars and underspecified representations*. Ph.D. Thesis, Universität Tübingen.

Lambek, J. (1958). The mathematics of sentence structure. *The American Mathematical Monthly, 65*, 154–169.

Le Roux, J. (2007). La coordination dans les grammaires d'interaction. Ph.D. Thesis, Université Nancy 2.

Le Roux, J., & Perrier, G. (2006). La coordination dans les grammaires d'interaction. *Traitement Automatique Des Langues, 47*(3), 89–113.

Marchand, J. (2006). *Algorithme de earley pour les grammaires d'interaction*. Université Nancy 2: Travaux universitaires.

Marchand, J., Guillaume, B., & Perrier, G. (2009). Analyse en dépendances à l'aide des grammaires d'interaction. In *16ième Conférence annuelle sur le Traitement Automatique des Langues Naturelles (TALN'09), Senlis, France*.

Marcus, M., Hindle, D., & Fleck, M. (1983). D-Theory: Talking about talking about trees. In *21st Annual meeting of the association for computational linguistics* (pp. 129–136).

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

Mouret, F. (2007). *Grammaire des constructions coordonnées. Coordinations simples et coordinations à redoublement en français contemporain*. Ph.D. Thesis, Université Paris 7.

Muskens, R., & Krahmer, E. (1998). Talking about trees and truth-conditions. In *Logical aspects of computational linguistics, LACL'98. Grenoble, France*.

Nasr, A. (1995). A formalism and a parser for lexicalised dependency grammars. In: *4th International workshop on parsing technologies, Prague, Czechoslowakia* (pp. 186–195), State University of NY Press.

Nivre, J. (2005). *Dependency grammar and dependency parsing*. MSI report 04071, Växjö University: School of Mathematics and Systems Engineering.

Perrier, G. (2000). Interaction grammars. In *18th International conference on computational linguistics, CoLing 2000, Sarrebrücken* (pp. 600–606).

Perrier, G. (2001). 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, Vol. 2250 of Lecture Notes in Artificial Intelligence. Havana, Cuba* (pp. 233–248).

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

Perrier, G. (2007). A French interaction grammar. In G. Angelova, K. Bontcheva, R. Mitkov, N. Nicolov, & K. Simov (Eds.), *RANLP 2007. Borovets Bulgarie* (pp. 463–467).

Pullum, G., & Scholz, B. (2001). On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In P. De Groote, G. Morrill, & C. Retoré (Eds.), *Logical aspects of computational linguistics, LACL 2001, Le Croisic, France, Vol. 2099 of lecture notes in computer science* (pp. 17–43), Springer.

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

Retoré, C. (2005). The logic of categorial grammars: Lecture notes. Technical Report RR-5703, INRIA.

Rogers, J., & Vijay-Shanker K. (1992). Reasoning with descriptions of trees. In *30th Annual meeting of the association for computational linguistics* (pp. 72–80).

Sag, I., Wasow, T., & Bender, E. (2003). Syntactic theory : A formal introduction. Center for the study of language and INF.

Sagot, B., Clément, L., de La Clergerie, E., & Boullier, P. (2006). The Lefff 2 syntactic lexicon for French: Architecture, acquisition, use. In *LREC 06, Genova, Italy*.

Stabler, E. (1997). Derivational minimalism. In C. Retoré (Ed.), *Logical aspects of computational linguistics, LACL'96, Nancy, France, Vol. 1328 of lecture notes in computer science* (pp. 68–95).

Steedman, M. (1985). Dependency and coordination in the grammar of Dutch and English. *Language, 61*(3), 523–568.

Steedman, M. (2000). *The syntactic process, Bradford books*. London: MIT Press.

Sygal, Y., & Wintner, S. (2009). Associative grammar combination operators for tree-based grammars. *Journal of Logic, Language and Information, 18*(3), 293–316.

Tesnière, L. (1934). Comment construire une syntaxe. *Bulletin de la Faculté des Lettres de Strasbourg 7*–12^{iéme} *année* (pp. 219–229).

Tesnière, L. (1959). *Eléments de syntaxe structurale*. Paris: Librairie C. Klincksieck.

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

Weir, D. (1988). *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. Thesis, University of Pennsylvania, Philadelphia, PA, USA.