Interaction Grammars

Guy Perrier
LORIA, Université Nancy2
BP 239
54506 Vandœuvre-lès-Nancy Cedex France
e-mail: perrier@loria.fr

Abstract

Interaction Grammars (IG) are a new linguistic formalism which is based on descriptions of under-specified trees in the framework of intuitionistic linear logic (ILL). Syntactic composition, which is expressed by deduction in linear logic, is controlled by a system of polarized features. In this way, parsing amounts to generating models of tree descriptions and it is implemented as a constraint satisfaction problem.

Introduction

IG can be presented as an attempt to bring together fundamental ideas, some coming from Tree Adjoining Grammars (TAG) and others from Categorial Grammars (CG), in order to overcome the specific limitations of each of these formalisms. The computational and linguistic relevance of TAG lies in its adjunction operation (Joshi et al., 1975; Kroch and Joshi, 1985) but the simplicity of its mechanism has a counterpart in the inflation of the lexicons that are required for expressing all grammatical phenomena of a language. Every time that a word is used in a new syntactic context, which can differ only by word order for example, a new elementary tree, which encodes this context, must be added to the lexicon in a direct manner or by means of a lexical rule. In this way, lexicons quickly become colossal, very awkward to use and very hard to maintain.
Recent works aim to solve this problem by factorizing linguistic information with the notions of *under-specified trees* and *tree descriptions*. These notions were introduced for TAG by Vijay-Shanker with the motivation of making adjunction monotone (Vijay-Shanker, 1992). Now, they are exploited fruitfully in various directions: for structuring TAG lexicons in hierarchical systems of modules (Candito, 1999) or for expressing semantic ambiguity (Muskens and Krahmer, 1998; Egg et al., 1998) for instance. At the same time, these notions are a way of relaxing the primitive adjunction operation in order to capture linguistic phenomena: such a possibility is exploited by (Rambow et al., 1995) within D-Tree Grammars and by (Kallmeyer, 1999) within Tree Description Grammars.
Unfortunately, the counterpart of a more flexible framework is often over-generation and a loss of computational efficiency in the absence of principles for controlling syntactic composition. By looking at CG, we can find some answers to this preoccupation: a fundamental idea of CG is that grammatical constituents are viewed as consumable resources (Retoré, 2000); these resources are divided into positive and negative resources which are complementary and search to neutralize themselves mutually. The core of CG is the Lambek Calculus (Lambek, 1958): by combining resource sensitivity and order sensitivity, this logic is a good candidate for representing the syntax of natural languages but at the same time, this combination entails a rigidity which limits its expressive power greatly. An appropriate way of relaxing this rigidity constitutes an important research area in CG (Moortgart, 1996).
The principle of IG is to combine the powerful notion of under-specified tree description linked to the TAG philosophy with control of syntactic composition by a system of polarities in accordance with the CG philosophy. More precisely, the basic objects of IG are *syntactic descriptions* which express dependencies between syntactic constituents in the shape of under-specified trees. Word order is referred to the same level as morphological information, which is represented by a system of features which are linked to the nodes of syntactic descriptions. Whereas a feature is usually a pair (attribute, value), an IG feature is a triplet (attribute, polarity, value) where polarity can take one of the three values -1, 0 or +1 and behaves like an electrostatic charge: for instance, a noun phrase which is waiting to receive a syntactic function in a sentence, carries a negative feature of type *funct* while a finite verb which is searching for its subject, carries a positive feature *funct* with value *subj*. Attraction between these dual features will make possible the fact that the verb finds its subject and, simultaneously, the noun phrase finds its function in the sentence. (Muskens and Krahmer, 1998) also recognized the necessity of introducing the notion of polarity in tree descrip-

tions as a mechanism for controlling syntactic composition; the difference with respect to IG lies in the granularity of the polarization which is finer in IG: in their proposal, the polarized objects are constituents, that is description nodes, whereas in IG one constituent can include several features with opposite polarities.

The framework which is chosen for representing syntactic descriptions in this paper is that of linear logic (Girard, 1987), more precisely a fragment of ILL (Lincoln, 1992). The resource sensitivity of linear logic allows one to express the fact that polarized features behave as consumable resources in IG: a positive feature has to find its dual feature once and only once . If we try to use classical or intuitionistic logic for modelling IG, the contraction and weakening rules, which are inherent in these logics, entail a loss of resource-sensitivity: for instance, a verb could take two subjects by appealing to the contraction rule and some noun phrases would not need to find their syntactic role in a sentence by appealing to the weakening rule. By discarding these two rules, linear logic provides a framework that exactly corresponds to the "electrostatic" laws that control polarized features.

In this framework, parsing takes the shape of logical deduction of *closed syntactic descriptions* from any syntactic descriptions: a description is said to be closed when it represents a completely specified syntactic tree where all features are neutralized.

If linear logic provides an elegant framework for representing IG, it gives no method for parsing efficiently and avoiding the combinatory explosion that may follow from the flexibility of the formalism. An answer to this problem is given by the paradigm of *constraint solving*. Parsing a phrase can be regarded as generating models of the partial description which is provided by a lexicon for the words of this phrase. The process is monotone and can be expressed as a *constraint satisfaction problem*. This constraint-based approach was inspired by the work of (Duchier and C., 1999; Duchier and Thater, 1999) on dominance constraints. (Blache, 1999) shows the advantages of such an approach both from a linguistic and computational viewpoint with the formalism that he proposes and he calls *Property Grammars*.

1 Syntactic descriptions as linear logic formulas

IG are formally defined as an ILL theory. Basic objects are *syntactic descriptions* which are represented by linear logic formulas in the following form:

$$Descr \quad ::= \quad Domin \mid Feat \mid Descr \otimes Descr \mid Descr \;\&\; Descr$$

If a syntactic description concerns the dominance relation between syntactic constituents, it has the type *Domin*; if it concerns the features which are used for characterizing syntactic or semantic properties of constituents, it has the type *Feat*. Finally, a description can be built recursively from two descriptions in two ways, which are expressed by the two linear logic conjunctions: the multiplicative tensor ($\otimes$) and the additive *with* (&).

1.1 Multiplicative and additive conjunction of resources in descriptions

A description $D_1 \otimes D_2$ requires all resources of both descriptions D_1 and D_2 while a description $D_1 \& D_2$ requires either the resources of D_1 or the resources of D_2 but not both. This use of the two linear logic conjunctions is consistent with their left introduction rules in the linear sequent calculus:

$$\frac{F_1, F_2, \Gamma \vdash G}{F_1 \otimes F_2, \Gamma \vdash G}\otimes_L \quad \frac{F_1, \Gamma \;\vdash G}{F_1 \& F_2, \Gamma \vdash G}\&_{L1} \quad \frac{F_2, \Gamma \vdash G}{F_1 \& F_2, \Gamma \vdash G}\&_{L2}$$

In this way, it is possible to describe all syntactic configurations of a word with a single lexical entry under the form of a syntactic description: common parts of these configurations are factorized whereas specific parts are distributed into alternations linked together with the connective *with*. For instance, a possible lexical entry for the finite verb *voit* in French has the shape $D_{voit} = D_1 \otimes (D_2 \& D3) \otimes (D_4 \& D5)$: D_1 contains information related to the subject which is common to all uses of the verb *voit*; D_2 expresses the canonical order subject-verb in the sentence that is headed by the verb *voit* whereas D_3 expresses the reverse order for which the subject must be realized under some conditions, such as in the phrase *Marie que voit Jean*; D_4 expresses that the verb has an explicit object whereas D_5 corresponds to circumstances where this object is not present, such as in the sentence *Jean voit*.

1.2 Under-specification of dominance between constituents

A description of type *Domin* has the following form:

$$Domin \quad ::= \quad Node > [LNodes] \mid Node > Node \mid Node >^* Node$$
$$LNodes \quad ::= \quad \epsilon \mid Node\ LNodes$$

A predicate $N > [N_1, \ldots, N_p]$ states that the constituent N is decomposed into the sub-constituents $N_1, \ldots, N_p$. The order between these sub-constituents is only used for identifying each one without any linguistic meaning; word order is dealt with at the same level as morphological information by means of features.

A predicate $N_1 > N_2$ expresses that N_2 is an immediate sub-constituent of N_1. Such a predicate is used when only partial information on the sub-constituents of a phrase is available.

A predicate $N_1 >^* N_2$ expresses that N_2 is embedded in N_1 at an undetermined depth. For instance, if we continue with description D_1 related to the verb *voit*, we can assume that it contains the formula

$(N_3 > [N_4, N_5]) \otimes (N_4 >^* N_6)$ which is interpreted as follows: the verb phrase N_3 is constituted of the verb N_4 and its object N_5; N_6 represents the bare verb whereas N_4 represents the verb which has been possibly modified by a clitic, a negation or an adverb. Under-specification of the dominance relation $N_4 >^* N_6$ leaves all these modifications open. Under-specification of dominance between constituents goes beyond TAG adjunction in that the nodes which are in a dominance relation do not necessarily have the same grammatical category and thus linguistic phenomena like wh-extraction can be expressed easily in this way.

1.3 Polarized and under-specified features

Descriptions of type *Feat*, related to features, have the following form:

$$\begin{array}{lll} Feat & ::= & Node : Attr\ Pol\ Val \mid \\ & & Var \in Dom \mid Var \notin Dom \\ Pol & ::= & \leftarrow \mid = \mid \rightarrow \\ Val & ::= & Const \mid Var \end{array}$$

A feature $Node : Attr\ Pol\ Val$ is a triplet composed of an attribute $Attr$, a polarity Pol and a value Val associated with a syntactic node $Node$. Usually, a feature is defined as a pair (attribute, value). In IG, we add a polarity to this pair so that features behave like electrostatic charges: a positive feature $Attr \rightarrow Val$ seeks a negative feature $Attr \leftarrow Val$ to neutralize it and conversely while a neutral feature $Attr = Val$ only acts as a filter through constraints on its value Val when it meets another feature of type $Attr$ at the same node.

In all cases, Val is either a constant which is selected from an infinite countable set $\mathcal{C}onst$ of feature values or a variable which is selected from an infinite countable set $\mathcal{V}ar$ of feature variables; then, its definition domain can be constrained by two kinds of predicates: $Val \in Dom$ and $Val \notin Dom$; Dom is a finite set of elements taken from $\mathcal{C}onst$.

Let us illustrate this presentation with a possible lexical entry for the proper noun *Jean*:

$$\begin{array}{lll} D_{Jean} & = & (N > [\,])\otimes \\ & & (N : cat = np) \otimes (N : funct \leftarrow V_1) \\ & & (N : ord \rightarrow 1) \otimes (N : phon =' Jean')\otimes \\ & & (N : gen = m) \otimes (N : num = sg)\otimes \\ & & (N : pers = 3) \end{array}$$

Some features are neutral by nature like agreement features: *gen=m* (gender=male), *num=sg* (number=singular), *pers=3* (person=3). Others are polarized by nature too: for instance, features of type *funct* which express syntactic functions. In the example above, the feature of type *funct* is negative because the noun phrase represented by N is waiting to receive a syntactic function (subject, object...); this function is not determined yet and thus it is represented by a variable V_1.

The phonological form of a constituent is determined by a system of two features: *phon* which gives the effective phonological form of the constituent and *ord* which gives the order in which its immediate sub-constituents must be concatened to build this phonological form. For instance, we find the formula $(N_1 : ord \rightarrow V_2) \otimes (V_2 \in \{12, 21\})$ in the description D_{voit} to express that the clause which has the verb *voit* as its head and is represented by node N_1 is a concatenation subject-verb phrase ($V_2 = 12$) or verb phrase-subject ($V_2 = 21$). When a node has no children, two cases occur: the node has an empty phonological form and the value of the feature *ord* is 0 or the node is a lexical anchor and the value of the feature *ord* is 1. In this case, the feature *phon* is used for retrieving the effective phonological form, which can be verified in the description D_{Jean}. Polarization of phonological forms expresses that some constituents are capable of giving a phonological form while others are waiting for one. As the previous examples shows, this polarity is not carried by the feature *phon* but by the feature *ord*. The interest of giving privilege to the feature *ord* with respect to the feature *phon*, is twofold: we can determine its value for a given node without being aware of the phonological form of the children, the effective phonological form will be rebuilt step by step from the leaves to the root of the final syntactic tree as soon as possible; another interest is that features of type *ord* can be dealt with like all other features; in particular, we can apply to them the same type of constraints.

Finally, it is interesting to mention that value sharing by different features is represented in an easy way by using a unique variable for the values of the concerned features.

2 Syntactic composition as deduction in a linear theory

By choosing a logical framework for a formal definition of IG, we find a natural way of expressing syntactic composition by means of deduction in linear logic according to the paradigm "parsing as deduction" of CG (for a broad survey of CG see (Retoré, 2000)). An interaction grammar is lexicalized in the sense that all linguistic resources are stored in a lexicon and these resources will be combined by using inference rules of the ILL deductive system for building the acceptable sentences of the corresponding language. Since syntactic descriptions use only a fragment of this logic and if we choose the framework of the sequent calculus, only seven ILL rules are useful:

$$\frac{}{F_1, \ldots, F_n \ \vdash \ F_1 \otimes \cdots \otimes F_n}id$$

$$\frac{\Gamma \ \vdash \ G}{1,\ \Gamma \ \vdash \ G}1_L \qquad \frac{F_1,\ F_2,\ \Gamma \ \vdash \ G}{F_1 \otimes F_2,\ \Gamma \ \vdash \ G}\otimes_L$$

$$\frac{F_1,\ \Gamma \ \vdash \ G}{F_1 \& F_2,\ \Gamma \ \vdash \ G}\&_{L1} \qquad \frac{F_2,\ \Gamma \ \vdash \ G}{F_1 \& F_2,\ \Gamma \ \vdash \ G}\&_{L2}$$

$$\frac{F[t/X],\ \Gamma \ \vdash \ G}{\forall\ X\ F,\ \Gamma \ \vdash \ G}\forall_L \qquad \frac{\Gamma_1 \ \vdash \ F \quad F,\ \Gamma_2 \ \vdash \ G}{\Gamma_1,\ \Gamma_2 \ \vdash \ G}cut$$

With respect to the usual presentation of the ILL sequent calculus (Lincoln, 1992), axiom *id* is defined a bit differently but this definition is equivalent to the original one for the logical fragment used by IG. Rule $\forall_L$ is a first order rule which is used here for instantiating a node variable with a concrete node or a feature variable with a concrete feature value.
Beside these general rules, we need proper axioms to express properties related to dominance relations, feature polarities, feature values and phonological forms. Concerning dominance relations, we have the following proper axiom schemes:

$$\overline{N_1 > N_2, (N_1 > [L, N_2, L']) \vdash (N_1 > [L, N_2, L'])}^{d_1}$$

$$\overline{N >^* N \vdash 1}^{d_2}$$

$$\overline{N_1 >^* N_3, (N_1 > [L, N_2, L']) \vdash N_2 >^* N_3 \otimes (N_1 > [L, N_2, L'])}^{d_3}$$

Axiom scheme d_1 expresses that immediate dominance is realized by a parent-children relation whereas axiom schemes d_2 and d_3 express that dominance is realized by finite sequences of parent-children relations (L and L*'* represent sequences of node variables).
The behaviour of polarities is represented by the following proper axiom schemes:

$$\overline{(N : A\ P\ V),\ (N : A = V)\ \vdash\ (N : A\ P\ V)}^{p_1}$$

$$\overline{(N : A \leftarrow V),\ (N : A \rightarrow V)\ \vdash\ (N : A = V)}^{p_2}$$

Properties related to feature domains and values are expressed by the following axiom schemes:

$$\overline{\vdash c \in \{c\} \cup D}^{v_1} \qquad \overline{\vdash c \notin D \backslash \{c\}}^{v_2}$$

In both axiom schemes, D represents a concrete finite set of feature values taken from $\mathcal{C}onst$ and $\cup$ and $\backslash$ represent the usual operations of union and difference of sets.
Finally, three axiom schemes are used for deducing the effective phonological form of a constituent from the order of the phonological forms of its children:

$$\overline{N > [\,], (N : ord = 0) \vdash N > [\,] \otimes (N : phon = `')}^{ph_1}$$

$$\overline{N > [\,], (N : ord = 1) \vdash N > [\,]}^{ph_2}$$

$$\overline{(N > [N_1, \ldots, N_p]), O, P_1 \vdash (N > [N_1, \ldots, N_p]) \otimes P_2}^{ph_3}$$

Schemes ph_1 and ph_2 respectively correspond to empty categories and lexical anchors.
In scheme ph_3, O is an abbreviation for $(N : ord = c(\sigma))$; σ is a permutation on $[\![1, p]\!]$ which expresses an order for concatenating the phonological forms $v_1, \ldots, v_p$ of the children nodes $N_1, \ldots, N_p$ of N and $c(\sigma)$ is a bijective encoding of this permutation with an integer. P_1 is an abbreviation for $(N_1 : phon = v_1), \ldots, (N_p : phon = v_p)$ and P_2 an abbreviation for the product $(N : phon = v_{\sigma(1)} \cdot \ldots \cdot v_{\sigma(p)}) \otimes (N_1 : phon = v_1) \otimes \cdots \otimes (N_p : phon = v_p)$.
A particular interaction grammar G is defined by its vocabulary $\mathcal{V}oc_G$ and by a lexicon $\mathcal{L}ex_G$; the vocabulary $\mathcal{V}oc_G$ includes the words used for building the language $\mathcal{L}_G$ generated by this grammar and the lexicon $\mathcal{L}ex_G$ associates a syntactic description to each word of $\mathcal{V}oc_G$. Now, we have to combine the resources provided by $\mathcal{L}ex_G$ by means of the inference rules and proper axioms of the linear theory $\mathcal{T}$ which has just been defined to compose well-formed and complete syntactic structures of G under the shape of *closed syntactic descriptions*. As a preliminary, we have to give a precise definition of a *closed syntactic description*:

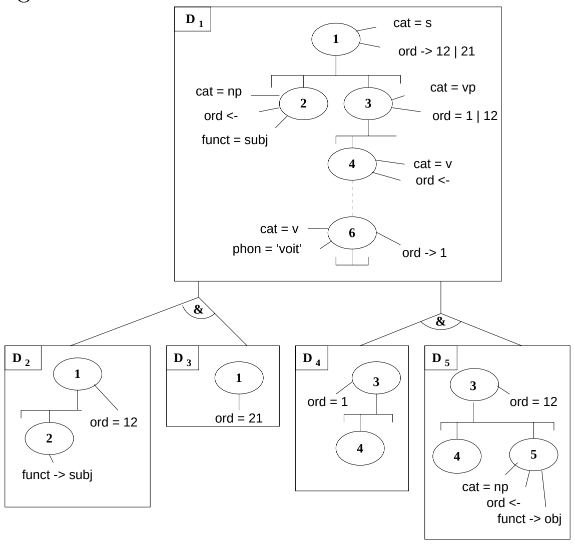
> *A closed syntactic description is a particular syntactic description in the shape* $S \otimes F$ *where* S *and* F*, respectively, represent the structural and feature parts of the description with the following conditions:*
>
> 1. S *is a product of predicates in the form* $(n > [n_1, \ldots, n_p])$*, where* n*,* n_1*, …,* n_p *represent concrete syntactic nodes, and the structure defined by all these parent-children relations is a tree;*
> 2. F *is a product of predicates in the form* $(n : attr = v)$*, where* n*,* $attr$ *and* v *represent concrete atoms, and for each pair* $(n, attr)$ *present in* F*, there is exactly one feature* $(n : attr = v)$ *in* F*.*
> 3. *For every syntactic node* n *in* S*, there is a feature* $(n : phon = v)$ *in* F*.*

Condition 1 guarantees that a closed syntactic description represents a completely specified tree. Condition 2 guarantees coherence and neutrality of the feature system which is attached at each syntactic node. Condition 3 guarantees the phonological well-formedness of the whole syntactic structure.
Now, let us explain how G generates closed syntactic descriptions from n lexical entries $D_{w_1}, \ldots, D_{w_n}$ corresponding to n words $w_1, \ldots, w_n$ taken from $\mathcal{V}oc_G$. For this, we need an additional description D_{root} to represent the root of the final syntactic tree which has the form:$(N_0 >^* N_1) \otimes \cdots \otimes (N_0 >^* N_p) \otimes (N_0 : ord \leftarrow V_0)$. Node N_0 represents the root of the syntactic tree and $N_1, \ldots, N_p$ are the nodes present in descriptions $D_{w_1}, \ldots, D_{w_n}$. Then:

> *A closed syntactic description* D *is said to be generated from the words* $w_1, \ldots, w_n$ *by grammar* G *if the sequent* $\forall \vec{N} \forall \vec{V}\ (D_{root} \otimes D_{w_1} \otimes \cdots \otimes D_{w_n}) \vdash D$ *is provable in the theory* $\mathcal{T}$ *(*$\vec{N}$ *and* $\vec{V}$ *represent all node variables and feature variables that are free in* D_{root}*,* $D_{w_1}, \ldots, D_{w_n}$*).*

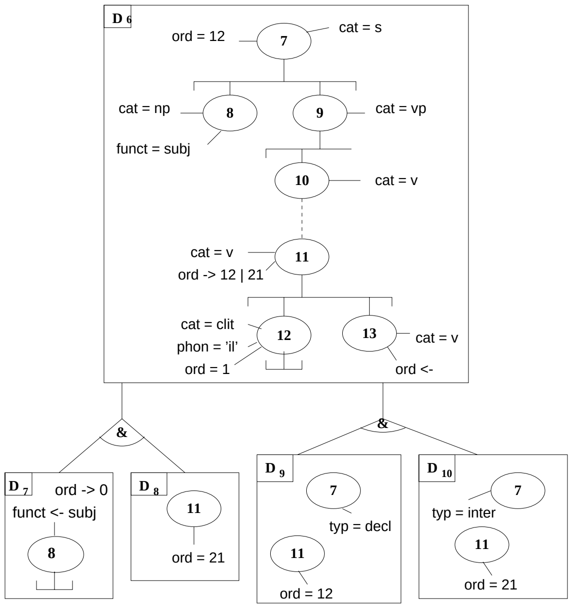
D describes a tree which represents the syntax of a phrase given by the feature *phon* of its root. If we add the predicate $(N_0 : phon = w_1 \ldots w_n)$ to D_{root}, we transform the generation of closed syntactic descriptions into parsing of the phrase $w_1 \ldots w_n$.

By continuing with the verb *voit*, let us give a very simple illustration of this mechanism. We assume that a lexicon provides us with three descriptions D_{voit}, D_{il} and D_{Jean} which respectively correspond to the finite verb *voit*, the personal pronoun *il* and the proper noun *Jean*. As it was described in subsection 1.1, D_{voit} has the shape $D_1 \otimes (D_2 \& D_3) \otimes (D_4 \& D_5)$ and it is schematized by the following diagram:



To remain readable, the diagram includes only the most significant features of every node. The notation $ord \rightarrow 12|21$ is an abbreviation for $ord \rightarrow V$ with $V \in \{12, 21\}$ and $ord \leftarrow$ means that the value of the feature *ord* is undetermined.

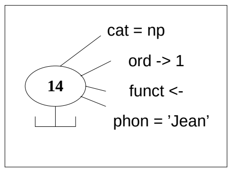
Description D_{il} has a structure that is similar to D_{voit}:



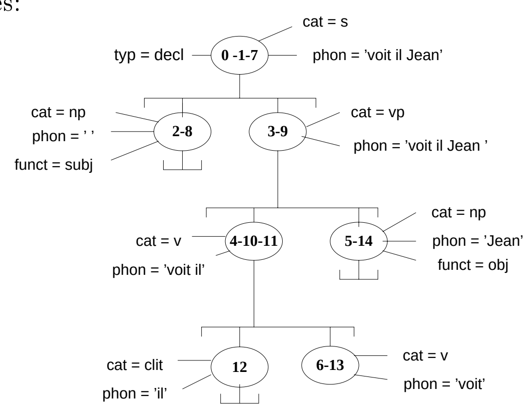
The first additive component of description D_{il}, $D_7 \& D_8$ represents a choice between the absence of an explicit subject in the sentence beside the personal pronoun *il* such as in the sentence *il voit Jean* and the presence of this subject such as in the sentence *Jean voit-il ?*. The second alternative entails that the sentence is interrogative if we ignore topicalization, which explains description D_8.

The second additive component of description D_{il}, $D_9 \& D_{10}$, represents a choice between the declarative type and the interrogative type of the sentence which depends on the relative order between the verb and the clitic.

Description D_{Jean} is reduced to the following single node:



From the description $\forall\ \vec{N}\ \forall\ \vec{V}\ (D_{root} \otimes D_{voit} \otimes D_{Jean} \otimes D_{il})$, it is possible to deduce three closed syntactic descriptions D_a, D_b and D_c, which respectively represent the syntax of the grammatical sentences :*il voit Jean*, *voit-il Jean ?* and *Jean voit-il ?*. In concrete terms, the deduction process that leads to these three solutions consists in plugging nodes of the initial descriptions with the aim of neutralizing all polarized features while respecting dominance and feature constrains. Let us detail the resulting description D_b by means of the syntactic tree it specifies:



The closed syntactic description that specifies the tree above represents the syntactic structure of the sentence *voit-il Jean ?*. The numbers that label its nodes are the traces of the nodes of the descriptions that have been plugged in the parsing process.

3 A constraint-based implementation

From the viewpoint of a computer scientist, a linguistic model has to show not only expressive power but also computational tractability. In the previous section, we have shown that IG computations reduce to ILL proofs. For the logical fragment that we consider here, three logical rules are a source of non-

determinism in proof search: $\&_{L1}$, $\&_{L2}$ and $\forall_L$. This takes the shape of three kinds of choice points in the parsing process: selecting the pertinent branch for every additive conjunction, identifying some node variables and instantiating feature variables in an appropriate manner. The NP-completeness of the implicative fragment of ILL (Kanovich, 1992) shows that it is hopeless to find a general parsing algorithm for IG that works in polynomial time in the worst cases. Experience has shown that, fortunately, these worst cases rarely occur in parsing natural languages. Nevertheless, the flexibility of IG entails a combinatory explosion of the parsing process if we use a "generate and test" method and leads us to choose a more appropriate method. The specification of our problem prompts us in a natural way to a constraint-based approach as it was suggested by some proposals for similar problems (Duchier and C., 1999; Duchier and Thater, 1999).
The problem can be formulated as follows:

> *Given a syntactic description D_0, find all closed syntactic descriptions D such that $\forall\vec{N}\,\forall\vec{V}\; D_0 \vdash D$ is provable in the theory $\mathcal{T}$ ($\vec{N}$ and $\vec{V}$ respectively represent the node variables $N_1, \ldots, N_n$ and the features variables $V_1, \ldots, V_m$ of D_0).*

A fundamental property of the deduction process that leads to a solution is monotonicity so that the problem can be expressed as a *constraint satisfaction problem (CSP)*. A CSP is specified from a set of variables to which constraints are applied. Here, we consider three sets of variables, which correspond to the three kinds of choice points in the parsing process:

1. the set $\{N_1, \ldots, N_n\}$ of syntactic node variables;
2. the set $\{V_1, \ldots, V_m\}$ of feature variables;
3. the set $\{S_1, \ldots, S_p\}$ of selection variables; every selection variable S_i is an integer variable which is associated with a connective $\&$ of D_0 and which is used for indicating the rank of the component of the correspondent additive conjunction that is selected in the deduction.

Selection and feature variables are considered as finite domain variables, which imply that all feature values are encoded as integers (one exception is that features of type *phon* remain strings).
Node variables are encoded indirectly via finite set variables by using the method proposed in (Duchier and C., 1999). Every node variable N_i is associated with five finite set variables $eq(i)$, $up(i)$, $down(i)$, $side(i)$ and $alt(i)$ which are used for locating the node i with respect to the others in the system of dominance relations. Because of the presence of additive conjunctions, a node i which is present in the description D_0 may be absent from a solution. In this case, $eq(i) = \{i\}$, $alt(i) = [\![1, n]\!] \backslash \{i\}$, $up(i) = down(i) = side(i) = \emptyset$; in the case that i is present in a solution, $alt(i)$ represents the nodes that are not selected in the solution whereas the selected nodes are distributed into the four sets $eq(i)$, $up(i)$, $down(i)$ and $side(i)$ according to their relative position with respect to i.
Constraints on the variables of the problem are divided into two parts:

- general constraints guarantee that the solutions D are effective closed syntactic descriptions;
- specific constraints guarantee that the solutions D are models of the initial description D_0.

3.1 General constraints

Treeness constraints For every node i, the partition of $[\![1, n]\!]$ between $eq(i)$, $up(i)$, $down(i)$, $side(i)$ and $alt(i)$ guarantees that the solution is a directed acyclic graph (DAG).
For expressing that all dominance relations which structure a solution must only be realized by parent-children relations, we must introduce constraints in which variables of type *eq(i)* and selection variables appear for expressing that every selected node variable must be identified with a node variable which is the parent in a selected parent-children relation.
In order to express that a solution is more than a DAG, that is a tree, we must add the following constraint: for every selected parent-children relation, the sets $down(j)$ for the children j present in this relation must be disjoint. Such a condition can be dropped if we want to extend the formalism to take into account resource sharing like coordination for instance; in this case, syntactic structures are no longer trees but DAGs.

Neutrality constraints Feature neutrality of a solution is guaranteed by constraints which also appeal to variables of type *eq(i)* and selection variables: for each attribute *Attr*, we consider two sets of sets in the shape *eq(i)*: the first corresponds to all selected predicates in the form $(Ni : Attr \leftarrow V)$ and the second to all selected predicates in the form $(Ni : Attr \rightarrow V)$. The elements of each of these sets must be disjoint sets and every element of the first set must be identified with one element of the second and conversely.
Other general constraints related to features and phonological forms are trivial.

3.2 Specific constraints

Such constraints are determined by D_0. Dominance constraints are easily implemented by combining selection variables and variables of type $eq(i)$, $up(i)$, $down(i)$, $side(i)$ (Duchier and Thater, 1999).

Feature constraints concern both feature variables and selection variables which are all finite domain variables so that their implementation appeals to classical tools in the domain of constraint programming.

3.3 A prototype parser for French

We have implemented a prototype parser for French. It is written in the language *Oz* (Smolka, 1995) which combines various aspects and modules, including constraint programming. Though the linguistic coverage of the lexicon is still limited, we have learnt lessons from the first experiments: in particular, neutrality constraints play a central role for restricting the search space, which confirms the importance of polarities for the computational efficiency.

Conclusion

Starting from TAG and CG, we have presented a linguistic formalism which aims at better capturing the flexibility of natural language by using two notions as its basis: *underspecification* and *polarities.* In some sense, they correspond to two important properties of natural language: ambiguity and resource sensitivity.

To regard parsing as a constraint satisfaction problem fits in with the flexibility of the formalism in terms of computational efficiency but, at the same time, it allows to go towards robustness beyond a traditional view of parsing in which only grammatical and completely specified structures are taken into account.

The success of IG does not essentially depend on the formal properties that are usually exhibited for grammatical formalisms: the characterization of the class of languages that are generated by these grammars or the complexity of general parsing algorithms. Formal properties matter but with respect to an essential goal: to extend the linguistic coverage of IG from toy lexicons to massive lexical databases. For this, IG have some advantages by making it easily to factorize and modularize information: such properties are decisive when one wants to extract information from a lexical database efficiently or to update data while maintaining the coherence of the whole base.

The success of IG will also depend on their capacity to integrate other linguistic levels than the syntactic level, the semantic level especially.

References

P. Blache. 1999. *"Contraintes et théories linguistiques : des Grammaires d'Unification aux Grammaires de Propriétés"*. Thèse d'Habilitation, Université Paris 7.

M.-H Candito. 1999. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien,.* Thèse de Doctorat, Université Paris 7.

D. Duchier and Gardent C. 1999. A constraint-based treatment of descriptions. In *IWCS-3, Jan 99,Tillburg, The Netherlands.*

D. Duchier and S. Thater. 1999. Parsing with tree descriptions: a constraint based approach. In *NLULP'99,Dec 99, Las Cruces, New Mexico.*

M. Egg, J. Niehren, P. Ruhrberg, and F. Xu. 1998. Constraints over lambda structures in semantic underspecification. In *COLING/ACL'98, Aug 98, Montreal, Quebec, Canada.*

J.-Y. Girard. 1987. Linear logic. *Theoretical Computer Science*, 50(1):1–102.

A. K. Joshi, L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

L. Kallmeyer. 1999. *Tree Description Grammars and Underspecified Representations.* Ph.D. thesis, Universität Tübingen.

M. Kanovich. 1992. Horn programming in linear logic is NP-complete. In *LICS'92, Jun 92, Santa Cruz, California*, pages 200–210.

A. Kroch and A. Joshi. 1985. Linguistic relevance of tree adjoining grammars. Technical Report MS-CI-85-18, Department of Computer and Information Science, University of Pennsylvania.

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

P. Lincoln. 1992. *Computational aspects of linear logic.* Ph.D. thesis, Stanford University.

M. Moortgart. 1996. Categorial Type Logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier.

R. Muskens and E. Krahmer. 1998. Talking about trees and truth-conditions. In *LACL'98. Dec 98, Grenoble, France.*

O. Rambow, K. Vijay-Shanker, and D. Weir. 1995. D-tree grammars. In *ACL'95*, pages 151–158.

C. Retoré. 2000. Systèmes déductifs et traitement des langues:un panorama des grammaires catégorielles. Research Report RR-3917, INRIA. To appear in Technique et Science Informatiques.

Gert Smolka. 1995. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin.

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